Spektrales Clustern und seine Anwendung auf Microarray–Daten

Diplomarbeit

vorgelegt von
Franz–Georg Ulmer
Matrikelnummer: 44816
Traithenstrasse 12
83115 Neubeuern

Erstkorrektor: Prof. Dr. Gunter Ritter
Zweitkorrektor: Prof. Dr. Thomas Müller–Gronbach

Universität Passau
Fakultät für Informatik und Mathematik

Inhaltsverzeichnis

Abbildungsverzeichnis **VI**

Tabellenverzeichnis **VIII**

1. Einleitung **1**
1.1. Was ist clustern? . . . 1
1.1.1. Die Semantik . . . 1
1.1.2. Die Clusterkriterien . . . 2
1.1.3. Die Optimierung . . . 2
1.1.4. Die Überanpassung . . . 3
1.2. Clustern versus Klassifikation . . . 3
1.3. Unser Anwendungsgebiet . . . 6
1.4. Die Merkmalsselektion . . . 6

2. Die Merkmalstransformation **7**
2.1. Wichtige Definitionen . . . 7
2.1.1. Die Metriken . . . 7
2.1.2. Die Unähnlichkeiten . . . 7
2.1.3. Die Ähnlichkeiten . . . 8
2.1.4. Der Graph . . . 8
2.2. Die Abstufung . . . 9
2.3. Die Transformation . . . 10
2.4. Unser Vorgehen bei Microarray–Daten . . . 12
2.5. Die Zusammenfassung . . . 13

3. Die Bewertung von Partitionen **14**
3.1. Die Clusterkriterien . . . 14
3.2. Wichtige Definitionen und Vorbereitungen . . . 14
3.2.1. Die Partition . . . 15
3.2.2. Der cut . . . 15
3.2.3. Der Grad eines Knotens . . . 16
3.2.4. Das Volumen eines Clusters . . . 16
3.2.5. Der zusammenhängende Graph . . . 17
3.3. Die Ideen der Clusterkriterien . . . 17

3.3.1. Das Vorgehen bei unseren Beispielen . . . 18
3.4. Der RatioCut . . . 19
3.5. Der NormalCut . . . 20
3.6. Der MinMaxCut . . . 22
3.7. Die Cluster Balance . . . 23
3.8. Die Effektivität bei nahezu optimalen Ähnlichkeiten . . . 24
3.9. Zusammenfassung der Kriterien . . . 27

4. Allgemeines Problem beim Clustern und seine Optimierung 28
4.1. Die Potenzmenge und Partition . . . 28
4.2. Die partielle Ordnung der Potenzmenge . . . 29
4.3. Der Bitvektor und die Potenzmenge . . . 30
4.4. Der optimale Durchlauf durch einen Bitvektoren . . . 30
4.5. Die Implementierungsdetails . . . 36
4.6. Die Zusammenfassung . . . 37

5. Spektrales Clustern 38
5.1. Wichtige Definitionen und Sätze . . . 38
5.1.1. Der ungerichtete, positiv gewichtete Graph der Ähnlichkeitsmatrix 38
5.1.2. Die Gewichtsmatrix des Graphen . . . 40
5.1.3. Die Degreematrix . . . 40
5.1.4. Die Laplacematrix . . . 41
5.1.5. Die normalisierte Laplacematrix . . . 42
5.1.5.1. Die normalisierte, symmetrische Laplacematrix . . . 42
5.1.5.2. Die Irrfahrt . . . 43
5.1.5.3. Der Zusammenhang von $\boldsymbol{L}_{\text{sym}}$ und $\boldsymbol{L}_{\text{rw}}$ . . . 44
5.1.6. Der Rayleigh–Quotient . . . 45
5.1.6.1. Die Diagonalmatrizen, Eigenvektoren und Eigenwerte . . 45
5.1.6.2. Die symmetrischen Matrizen, Eigenvektoren und Eigenwerte . . . 46
5.1.7. Der verallgemeinerte Rayleigh–Quotient . . . 46
5.2. Der RatioCut und die Laplacematrix . . . 47
5.2.1. Der allgemeine Fall . . . 47
5.2.2. Der Spezialfall für zwei Cluster . . . 48
5.3. Der NormalCut und die normalisierten Laplacematrizen . . . 49
5.3.1. Der allgemeine Fall . . . 50
5.3.2. Der Spezialfall für zwei Cluster . . . 50
5.4. Der MinMaxCut und die Adjazenzmatrix . . . 51
5.5. Der Zusammenhang mit Eigenwerten und Eigenvektoren . . . 52
5.5.1. Relaxation der Bedingungen . . . 52
5.5.1.1. Der RatioCut . . . 52
5.5.1.2. Der NormalCut . . . 53

5.5.1.3. Der MinMaxCut . . . 53
5.6. Das Gewinnen der Partition aus den Eigenvektoren . . . 54
5.6.1. Der 2–Clusterfall . . . 54
5.6.2. Der Simplex . . . 54
5.7. Die Zwischenbilanz . . . 55
5.8. Weitere Zugänge . . . 55
5.8.1. Die Irrfahrt . . . 55
5.8.2. Die Transposition . . . 58
5.9. Die Eigenvektoren im Optimalfall . . . 59
5.9.1. Der RatioCut . . . 59
5.9.2. Der NormalCut . . . 61
5.9.3. Der MinMaxCut . . . 61
5.10. Die Perturbationstheorie . . . 61
5.11. Die Komplexitätsabschätzung . . . 63
5.12. Die Fehlerabschätzung . . . 63
5.13. Exakt versus spektral . . . 64
5.13.1. Der euklidische Abstand als Ähnlichkeit . . . 64
5.13.2. Der graphentheoretische Zugang . . . 64
5.13.3. Der Vergleich . . . 64
5.14. Die Zusammenfassung . . . 67

6. Die Anzahlbestimmung der Cluster 68
6.1. Die allgemeinen Verfahren . . . 68
6.1.1. Das natürliche Clusterverhalten . . . 68
6.1.2. Die Stabilität (Weglassen von Daten) . . . 69
6.2. Die Eigenwerte . . . 70

7. Die Microarray–Daten 71
7.1. Die Grundlagen über DNA und RNA (Merkmale) . . . 71
7.2. Was sind Microarrays . . . 72
7.3. Wozu sind Microarrays gut . . . 73
7.4. Das spektrale Clustern von Microarray–Daten (Ähnlichkeiten) . . . 73
7.4.1. Die Gaußsche Glockenkurve . . . 74
7.4.2. Der Datensatz Golub . . . 76
7.4.3. Die Datensatzsammlung Stegmaier Leukemia . . . 78
7.4.3.1. Myeloid APL compound . . . 78
7.4.3.2. Myeloid primarycells . . . 79
7.4.3.3. Myeloid HL–60 undiff PMA ATRA . . . 81
7.5. Die alternierende Erweiterung . . . 81
7.5.1. Bagging . . . 82
7.5.2. Das Bagging basierte Verfahren . . . 82
7.5.2.1. Die Anwendung des Verfahrens auf den Datensatz Golub 85

7.5.2.2. Die Anwendung auf Myeloid HL–60 undiff PMA ATRA 87
7.6. Die Überanpassung 88
7.7. Der Vergleich des Ergebnisses mit anderen Arbeiten 88
7.8. Die Zusammenfassung 89

8. Die Schlussbetrachtung **90**

Literaturverzeichnis **IX**

A. Ein kleines Gedankenexperiment **XI**

Abbildungsverzeichnis

1.1. Einfaches Clusterbeispiel 4
1.2. Semantik–abhängiges clustern. Punkte die auf einer Geraden liegen gehören zusammen. 4
1.3. Semantik–abhängiges clustern. Trennung durch die x–Achse. 5
1.4. Semantik–abhängiges clustern. Trennung durch die y–Achse. 5

2.1. Beispiel eines gerichteten, positiv gewichteten Graphens 8
2.2. Beispiel eines ungerichteten, positiv gewichteten Graphens 9
2.3. Abstufung der Definitionen 9
2.4. y-Merkmal dominiert x-Merkmal. 11
2.5. x–Merkmal dominiert y–Merkmal. 11
2.6. Beispiel von drei Datenpunkten in $\mathbb{R}$ 12
2.7. Beispiel einer Unähnlichkeit als ungerichteter, positiv gewichteter Graph dargestellt (aus Abb. 2.6 gewonnen) 12
2.8. Beispiel einer Ähnlichkeit als ungerichteter, positiv gewichteter Graph dargestellt (aus Abb. 2.7 gewonnen) 12
2.9. Resultierender ungerichteter, positiv gewichteter Graph (aus Abb. 2.8 gewonnen) 12

3.1. Summe roter Kanten entspricht dem cut(Cluster1, Cluster2). Kantengewichte in der Darstellung weggelassen. 16
3.2. Summe roter Kanten entspricht d_i. Kantengewichte in der Darstellung weggelassen. 16
3.3. Unzusammenhängender Graph 17
3.4. $g(x) = g_{c_i}$ mit $c_i = 1, a_i = x$ und somit $b_i = 1 - x$ 18
3.5. Richtiges clustern mit Hilfe des RatioCut. 19
3.6. Schlechtes clustern mit Hilfe des RatioCut. 20
3.7. Richtiges clustern dank NormalCut. 21
3.8. Ein Punkt ist falsch geclustert von RatioCut. 21
3.9. Bessere, aber nicht perfekte Partition durch *MinMaxCut* (quadratischer euklidischer Abstand wurde verwendet). 22
3.10. *NormalCut* Clustert zwei Datenpunkte falsch (quadratischer euklidischer Abstand wurde verwendet). 23

3.11. Ungerichteter, positiv gewichteter Graph mit jeweils acht Datenpunkten innerhalb eines Clusters. Die schwarzen Kanten entsprechen einer Ähnlichkeit von 1, während die grauen einer Ähnlichkeit von $0,5$ entsprechen. 25

4.1. Generierter Baum des FRTPS Algorithmus. Er wird in der Tiefe und dann von links nach rechts durchlaufen. 34
4.2. $n \to n+1$ 34
4.3. Der Reihe nach erzeugte Bitvektoren. Durchgehend von oben und unten, sind die Elemente jeweils das Komplement. 35

5.1. Ungerichteter, positiv gewichteter Ausgangsgraph 39
5.2. 1–gerichteter, positiv gewichteter Nachbarschaftsgraph entstanden aus Abbildung 5.1 40
5.3. mutual 1–nearest Neighboorhood Graph 40
5.4. 1–nearest Neighboorhood Graph 40
5.5. Vereinfacht dargestellte Irrfahrt 58
5.6. Gruppierte Irrfahrt mit zwei neuen Zuständen 58
5.7. cockroach Graph aus Luxburg [13] 63
5.8. Spektrale Ergebnisse mit ihren Eigenvektoren 66

6.1. Drei Cluster wobei zwei ähnlich zueinander sind (Clusterübergreifende Kanten weggelassen). 69
6.2. Zwei Cluster die in drei Cluster unterteilt werden (Kanten weggelassen). Jede Farbe entspricht einem Cluster. 69
6.3. (s. Luxburg [13]) Drei Datensätze und die kleinsten 10 Eigenwerte bezüglich L_{rw}. 70

7.1. DNA vs. RNA [2] 72
7.2. $w(x) = \left(\frac{1}{1+x}\right)^k$ 74
7.3. $s(x) = \exp(-(\frac{1}{2k^2})x^2)$ 76
7.4. Die Eigenwerte bezüglich L_{sym} 77
7.5. Die Eigenwerte bezüglich L_{sym} 79
7.6. Die Eigenwerte bezüglich L_{sym} 80
7.7. Die Eigenwerte bezüglich L_{sym} 81
7.8. Dichtefunktionen zweier stark überlappender Verteilungen 83
7.9. Dichtefunktionen zweier kaum überlappender Verteilungen 83
7.10. Ablaufdiagramm des Bagging basierten Verfahrens. 85
7.11. Die Partitionen der Väter zu Ihren Söhnen unterscheiden sich um einen Punkt. 87

Tabellenverzeichnis

3.1. Überblick aller Kriteriumswerte für $|C_0| = |C_1| = 15$ 26
3.2. Überblick aller Kriteriumswerte für $|C_0| = 22, |C_1| = 8$ 26
3.3. Überblick aller Kriteriumswerte für $|C_0| = 26, |C_1| = 4$ 26

7.1. Resultate im Überblick für zwei Cluster (Golub) 77
7.2. Resultate im Überblick für drei Cluster (Golub) 77
7.3. Abkürzungen 78
7.4. Resultate im Überblick für zwei Cluster (Myeloid APL compound) 78
7.5. Resultate im Überblick für drei Cluster (Myeloid APL compound) 79
7.6. Resultate im Überblick für zwei Cluster (Myeloid primarycells) 80
7.7. Resultate im Überblick für drei Cluster (Myeloid primarycells) 80
7.8. Resultate im Überblick für zwei Cluster (Myeloid HL–60 undiff PMA ATRA) 81
7.9. Vergleich der Resultate vom Datensatz Golub für zwei Cluster 89
7.10. Vergleich der Resultate vom Datensatz Golub für drei Cluster 89

Kapitel 1.

Einleitung

1.1. Was ist clustern?

Umgangssprachlich ist ein Cluster eine Teilmenge von Objekten. Der Vorgang der Clusterung bedeutet eine ungruppierte Menge in Cluster zu unterteilen. Betrachtet man einen zweidimensionalen Bereich, so könnte man meinen, dass Clustern nichts anderes als das Zeichnen von Kreisen ist (Abb. 1.1). Dies reicht oft tatsächlich aus und wird meistens mit dem *k–means* Algorithmus erreicht (s. Ritter [21]).
In Wirklichkeit versucht man eine Menge in gegenseitig isolierte, in sich kohärente Teilmengen, zu zerlegen. Wie man in den Abbildungen 1.2, 1.3 und 1.4 sehen kann gibt es nicht immer eine klare Partition. Das Clustern ist eine Art der Interpretation von Ursprungsdaten und ist somit von der Semantik der Daten abhängig. Da diese bis heute in keine mathematische Sprache gefasst werden konnte, stellt sich die Frage, ob es je eine umfassende Arbeit über Clusterung geben wird.

1.1.1. Die Semantik

Hinter einem Cluster verbirgt sich im besten Fall eine Semantik, da eine Partition ähnliche Objekte (in Cluster) bündelt. Als Beispiel könnte man die Menschen einmal der Größe nach clustern, dann hätte ein Cluster zum Beispiel die Semantik „groß“ und ein anderes die Semantik „klein“. Beinhaltet nur **ein** Cluster Datenpunkte, die nichts gemein haben, so ist die Semantik dieses Clusters „der Rest“, da die restlichen Cluster einfach als Sieb zu interpretieren sind. Haben mehr als **zwei** Cluster Datenpunkte in sich, die nichts gemein haben, so handelt es sich um eine misslungene Partition. Denn entweder tragen die Daten keine Information in sich, beziehungsweise wurden in zu viele Gruppen (Clustern) unterteilt, oder die Ähnlichkeit zwischen Datenpunkten wurde falsch definiert. Im Allgemeinen ist das grundlegende Problem des Clusterns, dass die Semantik hinter den Clustern nicht bekannt ist. Im Gegensatz dazu gibt es die Klassifikation, bei der bereits im voraus klar ist in welche Gruppen man unterteilt, und somit über die

Semantik bescheid weiß.
Bei unserem Anwendungsgebiet geht es darum, wie man mehr über eine Krankheit in Erfahrung bringt. Kann man Sie unterteilen und somit gezielter behandeln? Oder anders ausgedrückt: Gibt es eine feinere Semantik einer Krankheit?
Hinter einer Datenmenge verstecken sich nicht immer klare natürliche Cluster. Betrachtet man die Abbildungen 1.2, 1.3 und 1.4, so könnte man zwei Cluster durch eine Trennung der x-Achse, der y-Achse, oder einer Trennung durch die Geraden (alle Punkte auf einer Geraden gehören zu einem Cluster) gewinnen. Welche Partition erwünscht ist, versteckt sich eben gerade hinter der Codierung der Datenpunkte und der gewünschten Semantik.
Die Ähnlichkeit von Datenpunkten beeinflusst die Semantik der Partitionen wodurch sich die Semantik hinter den Ähnlichkeiten verbirgt. Des weiteren können durchaus mehrere Semantiken hinter der gleichen Partition stecken.

1.1.2. Die Clusterkriterien

Um Partitionen miteinander vergleichen zu können benötigt man ein Kriterium, welches uns ermöglicht, eine Bewertung einer Partition zu treffen. Dabei gibt es zunächst einmal die heuristischen Kriterien (s. single Linkage, complete Linkage usw. in Ritter [21]), die statistischen Modelle (betrachtet die Datenpunkte als Realisierung eines stochastischen Prozesses wie z.B. ML und MAP Schätzer s. Ritter [21]) und die Goodness–of–fit Kriterien (s. Toma [27]). Die von uns betrachteten Clusterkriterien *RatioCut, NormalCut* und *MinMaxCut* gehören zu den heuristischen Kriterien.

1.1.3. Die Optimierung

Durch die Clusterkriterien ist man in der Lage Partitionen zu bewerten und so von einer optimalen Partition zu sprechen (Eindeutigkeit gilt nicht immer). Diese optimale Partition zu finden ist jedoch sehr aufwendig, da es zu viele Möglichkeiten gibt eine Partition zu wählen. Betrachtet man alle surjektiven Abbildungen $N \to K$, wobei $N := \{1 \dots n\}$ (Menge aller Datenpunkte), $K := \{1 \dots k\}$ (Menge aller Cluster) ist, so gibt es von ihnen $\sum_{j=0}^{n}(-1)^j \binom{n}{j}(n-j)^k$ viele (s. Ritter [19]). Da man eine Partition durch eine surjektive Funktion darstellen kann, weiß man über eine obere Schranke bescheid. Der Name eines Clusters ist bei einer Partition nicht von Bedeutung, weshalb verschiedene surjektive Funktionen die gleiche Partition darstellen. Es gibt $k!$ viele verschiedene Bijektionen von $K \to K$ (s. Ritter [19]), und somit $k!$ viele verschiedene Bezeichnungen für die selbe Partition. Zusammengefasst gibt es also $\frac{\sum_{j=0}^{n}(-1)^j \binom{n}{j}(n-j)^k}{k!}$ verschiedene Partitionen, wodurch die Suche nach der optimalen Partition im allgemeinen NP–hart (s. Schöning [23]) ist, weshalb eine Approximation notwendig wird.

Wie der Name des spektralen Clusterns schon verrät, werden Eigensysteme herangezogen, um eine approximierte Lösung zu erhalten (s. Luxburg [13]). Genauer gesagt kann durch Eigensysteme ein geeigneter Startpunkt gefunden werden, um somit eine lokale Suche nach einer optimalen Partition durchzuführen.
Zur Überprüfung der Ergebnisse der Approximation, werde ich einen exakten Algorithmus vorstellen, der trotz Optimierung exponentielle Laufzeit hat. Die Problemgröße bei einem heutigen Rechner ist wegen der exponentiellen Laufzeit auf circa 35 Datenpunkte beschränkt.

1.1.4. Die Überanpassung

Um ein passendes Modell für ein allgemeines Problem zu finden, wird oft ein konkretes Problem betrachtet, dessen Lösung man bereits kennt. Darauf hin wird ein Modell geschaffen, welches die Lösung liefert, und anschließend wird dieses verallgemeinert. Hierbei gibt es oft Probleme, da das Modell so getrimmt wurde, dass es zwar ein perfektes Ergebnis des speziellen Problems liefert, aber allgemein keine Bedeutung hat. Freie Parameter, die durch bloßes ausprobieren so eingestellt wurden, dass es zu einem gewünschten Ergebnis kam, weisen auf ein überangepasstes Modell hin.
Ich werde deshalb weitestgehend auf Modelle mit freien Parametern verzichten, oder diese zumindest automatisch schätzen lassen. Erst zum Schluss werden wir eine alternierende Erweiterung kennen lernen, wobei Parameter frei von Hand eingestellt werden müssen. Dabei ist zu beachten, dass die Parameter so eingestellt werden, dass es zu einem möglichst stabilen Ergebnis kommt (verstellt man einen Parameter, so darf das Ergebnis nicht stark abweichen (s. Sick [25])).

1.2. Clustern versus Klassifikation

Gehen wir nun auf die Unterschiede zwischen Klassifikation und Clusterung ein. Wie bereits erwähnt, liegt ein Unterschied in dem Wissen über die Semantik eines Clusters vor. Bei einer Klassifikation ist die Semantik immer gegeben, während dies bei dem Clustern nicht immer der Fall ist. Der tatsächliche Unterschied liegt im Modell, denn der Klassifikation liegt ein Modell zugrunde, welches bewusst gewählt wird, wobei Clustern dieses Modell mitschätzt. Das oben genannte Beispiel mit den Menschen wird hier nochmals aufgegriffen. Eine Klassifikation sagt aus, dass ein Mensch, der über 180 cm groß ist, als groß eingestuft wird. Bei dem Clustern hingegen wird dieser Schwellenwert mitgeschätzt. Auch die Clusteranzahl gehört oft zu dem Modell, welches mitgeschätzt werden muss. Bei der Klassifikation geht man von kleinen und großen Menschen aus, wohingegen mit dem Clustern untersucht werden kann, ob die Daten (die Größe der Menschen) eine feinere Unterteilung in sich tragen (gibt es bezüglich der Größen irgendwelche Sprünge).

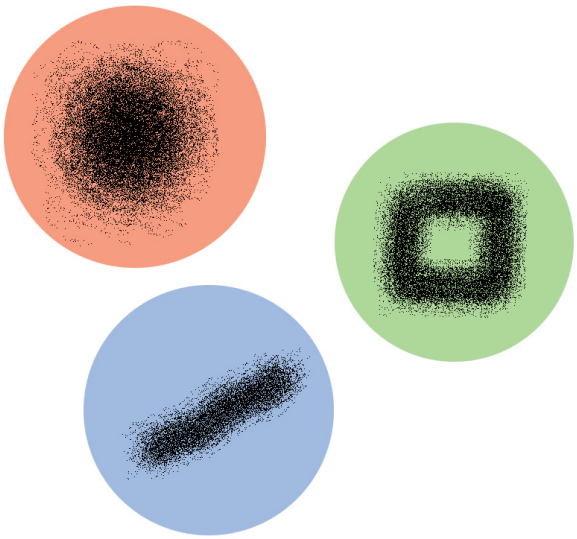

Abbildung 1.1.: Einfaches Clusterbeispiel

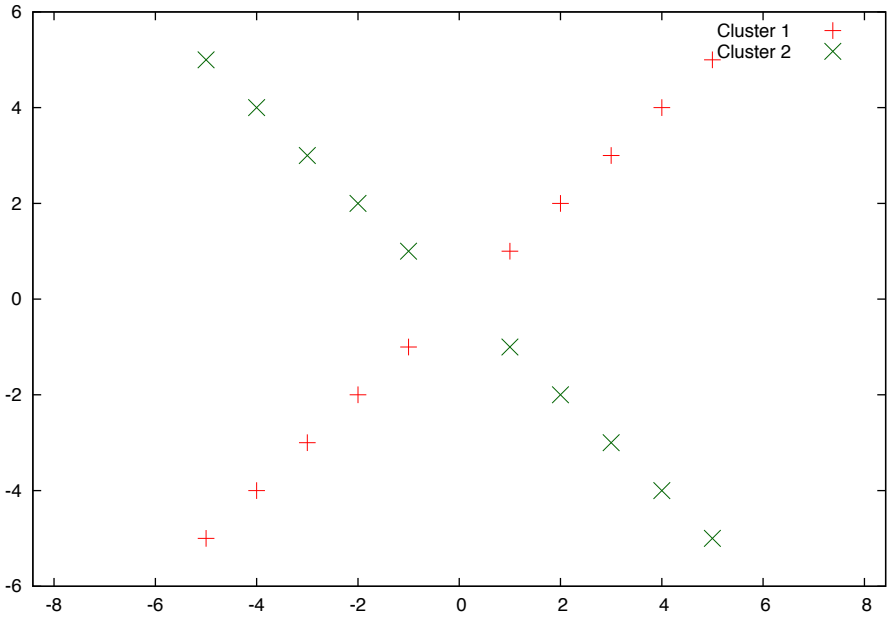

Abbildung 1.2.: Semantik–abhängiges clustern. Punkte die auf einer Geraden liegen gehören zusammen.

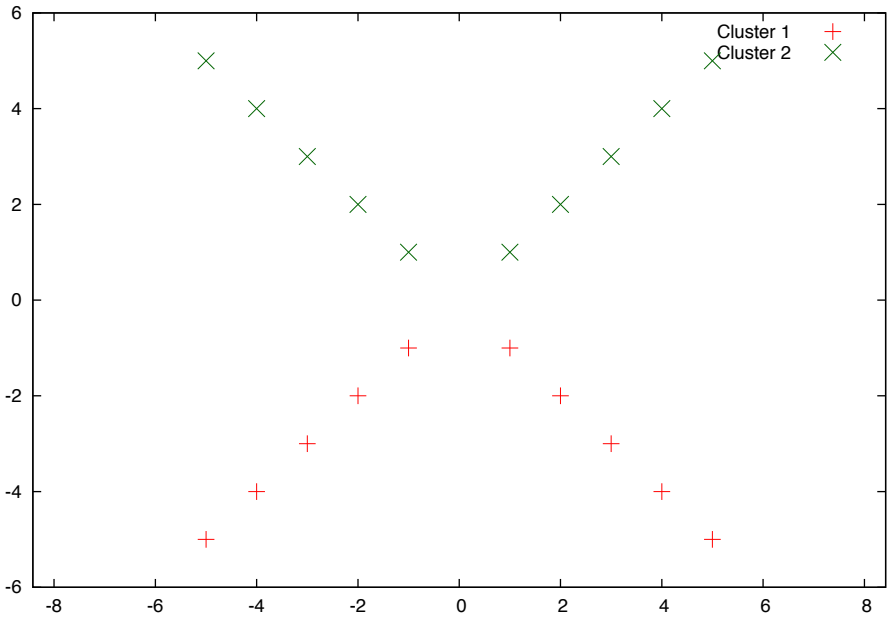

Abbildung 1.3.: Semantik–abhängiges clustern. Trennung durch die x–Achse.

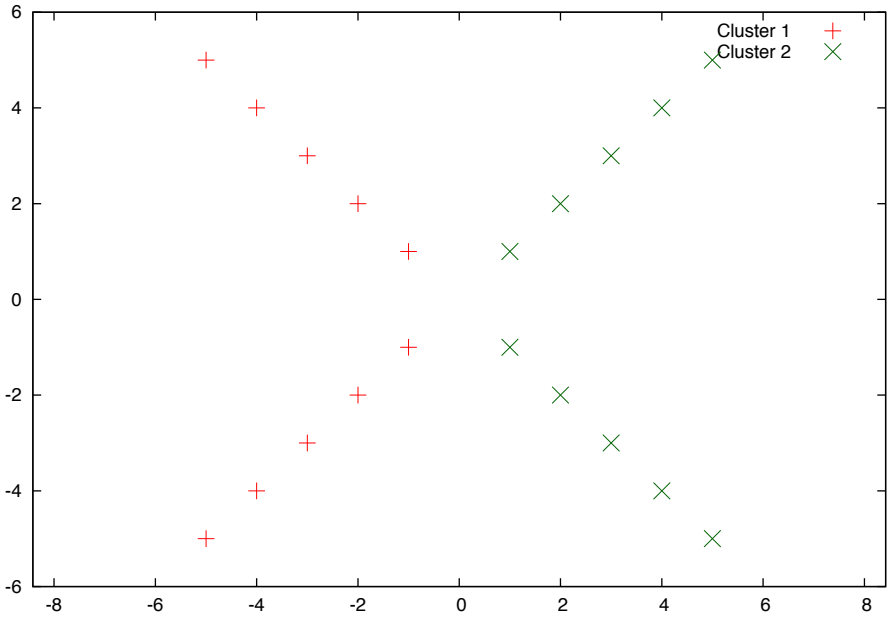

Abbildung 1.4.: Semantik–abhängiges clustern. Trennung durch die y–Achse.

1.3. Unser Anwendungsgebiet

Als Anwendungsgebiet werden wir uns den *Microarray–Daten* widmen. Es handelt sich hierbei um Daten die von kleinen Chips kommen, die die RNA analysieren. Auf diesen Chips werden die Bestandteile einer RNA durch Farben dargestellt, was später noch genauer erläutert wird. Bei dem betrachteten Datensatz handelt es sich um den Golub et al. Datensatz [8], welcher Krebspatienten analysiert. Wir werden diese Daten auf Gemeinsamkeiten untersuchen, womit man in der Lage wäre eine Risikovorhersage zu treffen und die Krankheit zu untergliedern. Des Weiteren wird die Stegmaier et al. [26] Datensatzsammlung betrachtet. Sie beinhaltet drei verschiedene Datensätze mit unterschiedlich vielen Datenpunkten und Clustern. Ein Großteil der Daten stammt von Patienten, die an Leukemie erkrankt sind.

1.4. Die Merkmalsselektion

Bei den Microarray–Daten handelt es sich um hochdimensionale Datensätze (> 7000). Nur ein geringer Bruchteil dieser Dimensionen hat einen Zusammenhang mit einer Krankheit, weshalb es durch die anderen Dimensionen zu Störungen kommen kann. Es ist also von Vorteil die wichtigen Merkmale zu selektieren, oder zumindest die störenden auszusortieren. Für diesen Zweck wurde von mir eine alternierende Erweiterung entwickelt, die auf dem Grundsatz des *Bagging* beruht. Dabei werden Stichproben von Merkmalen selektiert und die entsprechenden Partitionen berechnet. Diese werden zum Schluss auf ihre Gültigkeit überprüft, die entsprechenden Merkmale zu einer finalen Menge zusammengefasst, und die dazugehörige Partition wird ein letztes mal berechnet.

Kapitel 2.

Die Merkmalstransformation

Unter einer Merkmalstransformation versteht man den Übergang von den Merkmalen zu einer Ähnlichkeit oder einem ungerichteten, positiv gewichteten Graphen. Diese Merkmalstransformation beeinflusst gerade die Semantik von Clustern weshalb zunächst einige Definitionen benötigt werden.

2.1. Wichtige Definitionen

In diesem Abschnitt werden einige Definitionen eingeführt und aufgezeigt, dass diese miteinander verwandt sind. Hierbei handelt es sich um Metriken, Unähnlichkeiten, Ähnlichkeiten und Graphen.

2.1.1. Die Metriken

Eine Metrik ist eine Funktion zum Messen von Längen. Die Definition einer Metrik wird hier vorrausgesetzt und kann zum Beispiel in Heuser [11] nachgeschlagen werden.

2.1.2. Die Unähnlichkeiten

Definition 2.1 *(s. Ritter [21]) Sei M eine beliebige Menge. Eine Abbildung $u : M \times M \to [0, \infty]$ heißt* ***Unähnlichkeit*** *auf M, wenn gilt:*

(i) $\forall_{x \in M} u(x, x) = 0$

(ii) $\forall_{x,y \in M} u(x, y) = u(y, x)$.

Umgangssprachlich bedeutet dies, dass die Reihenfolge bedeutungslos ist und alle Objekte zu sich selbst nicht unähnlich sind.

2.1.3. Die Ähnlichkeiten

Definition 2.2 *(s. Ritter [21]) Sei $s_0 \in]0, \infty[$ und M eine beliebige Menge. Eine Abbildung $s : M \times M \to [0, s_0]$ heißt **Ähnlichkeit** auf M, wenn gilt:*

(i) $\forall_{x \in M} s(x, x) = s_0$

(ii) $\forall_{x,y \in M} s(x, y) = s(y, x)$.

Es liegt die gleiche Bedeutung wie bei Definition 2.1 vor, wobei hier alle Objekte zu sich selbst ähnlich sind.

2.1.4. Der Graph

Definition 2.3 *Ein **gerichteter, positiv gewichteter Graph** G ist ein Tripel (V, E, f), wobei:*

(i) $V \neq \emptyset$ *endliche Menge, die Elemente aus V heißen Knoten (oder Ecken)*

(ii) $E \subseteq \{(a, b) | a, b \in V\}$ *eine Menge von Tupeln. Ist $a \neq b$ so ist $(a, b) \neq (b, a)$. Die Elemente aus E heißen Kanten.*

(iii) $f : E \to \mathbb{R}_+$ *eine Abbildung von der Kantenmenge in die positiven reellen Zahlen, welche in der Literatur oft als Gewichtsfunktion bezeichnet wird.*

Umgangssprachlich ist ein gerichteter, positiv gewichteter Graph eine Ansammlung von Objekten, wobei zwischen den Objekten eine gerichtete Kante gezogen werden kann. Die gerichteten Kanten verfügen über ein Gewicht, welches eine positive reelle Zahl darstellt (siehe Abb. 2.1).

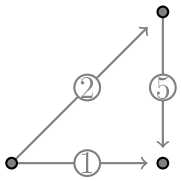

Abbildung 2.1.: Beispiel eines gerichteten, positiv gewichteten Graphens

Definition 2.4 *Ein **(ungerichteter), positiv gewichteter Graph** G ist ein Tripel (V, E, f), wobei:*

(i) $V \neq \emptyset$ *endliche Menge, die Elemente aus V heißen Knoten (oder Ecken)*

(ii) $E \subseteq \{\{a, b\} | a, b \in V\}$ *eine Menge von zwei– oder ein–elementigen Mengen. Die Elemente aus* E *heißen Kanten.*

(iii) $f : E \to \mathbb{R}_+$ *eine Abbildung von der Kantenmenge in die positiven reellen Zahlen, welche in der Literatur oft als Gewichtsfunktion bezeichnet wird.*

Anders als zu dem gerichteten, positiv gewichteten Graphen, lässt man die Richtung einer Kante aussen vor (siehe Abb. 2.2).

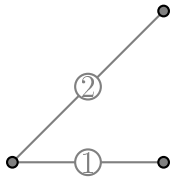

Abbildung 2.2.: Beispiel eines ungerichteten, positiv gewichteten Graphens

2.2. Die Abstufung

Es gibt eine Abstufung dieser Definitionen (s. Abb. 2.3). So kann man jede *Metrik* als *Unähnlichkeit* auffassen und jede *Unähnlichkeit* in eine *Ähnlichkeit* umwandeln, indem man s_0 größer oder gleich der maximalen Unähnlichkeit zweier Objekte wählt ($s(x, y) = s_0 - u(x, y)$). Zuletzt kann man jede *Ähnlichkeit* als *ungerichteten, positiv gewichteten Graphen* auffassen, indem man alle Objekte miteinander verbindet und jedes Einzelne zusätzlich auf sich selbst verweisen lässt. Die Kantengewichte entsprechen den Ähnlichkeiten zwischen den Objekten.
Zu beachten ist, dass nicht jeder ungerichteter, positiv gewichteter Graph als Ähnlichkeit aufgefasst werden kann, da Kanten fehlen können, und somit die Ähnlichkeit zwischen diesen Objekten undefiniert ist. Weiterhin ist nicht jede Unähnlichkeit eine Metrik, da man leicht die Dreiecksungleichung verletzen kann.

Metrik → Ähnlichkeit → ungerichteter, positiv gewichteter Graph

Abbildung 2.3.: Abstufung der Definitionen

2.3. Die Transformation

Betrachtet man n Menschen und versucht man diese in Gruppen zu unterteilen, also sie zu clustern. Jeder Mensch sei hierbei durch einen m-dimensionalen Vektor beschrieben, wie zum Beispiel Größe in cm, Geschlecht, Staatsangehörigkeit, Konfession, und so weiter. Dabei seien die Werte durch die natürlichen Zahlen kodiert, anders ausgedrückt: Die Menschen werden in $\mathbb{N}^m$ kodiert und gruppiert, die nahe beieinander liegen (eine Ähnlichkeit ist Definiert durch: $s(x, y) = \max_{u,v} d_{u,v} - d_{x,y}$ wobei $d_{i,j}$ der euklidische Abstand von Punkt i zu Punkt j ist). Wenn man weiterhin annimmt, dass bis auf die Größe alle Werte kleiner 10 sind, und diese Menschen nun in drei Cluster unterteilt, so würde man Folgendes erwarten: Man hätte die Menschen in die Gruppen klein, mittel und groß geclustert, da die Größe das dominierende Merkmal wäre (die Größe beeinflusst maßgeblich den Abstand und somit die Ähnlichkeit (s. Abb. 2.4 vs. Abb. 2.5)). Alle anderen Werte würden nur ein Rauschen verursachen, also keine großen Effekte erzielen. Im schlechtesten Fall kann dieses Rauschen sogar eine Falschzuordnung von einigen wenigen Datenpunkten bewirken.
Es ist also eine Vorverarbeitung der Daten notwendig, was in diesem Fall eine Projektion auf die Größe, und somit eine Merkmalsselektion bedeutet. Bei diesem Beispiel wird das Merkmal „Größe“ selektiert, wodurch ähnlich große Menschen gleichen Clustern zugeordnet werden. Durch diese Vorverarbeitung kann man nun auch andere Partitionen generieren, wie zum Beispiel die Gruppierung in Religionen. Dazu müsste nur das Religionsmerkmal betrachtet werden und die Kodierung so gewählt werden, dass ähnliche Religionen näher beieinander liegen als unähnliche. Betrachten man hier als Beispiel den Evangelismus und den Katholizismus, die sich näher stehen, als hierzu der Buddhismus. Man kann Beispielsweise sämtliche Gebote aller Religionen betrachten und jeder Religion einen entsprechenden Vektor zuordnen, wobei alle Einträge zwischen 1 (ist in dieser Religion wichtig) und 0 (ist nicht wichtig) liegen. Als Ähnlichkeit könnte man wieder die oben definierte verwenden ($s(x, y) = \max_{u,v} d_{u,v} - d_{x,y}$).
Zusammenfassend bleibt zu sagen, dass eine gewissenhafte Definition einer Ähnlichkeit von Datenpunkten von Nöten ist, um eine aussagekräftige Partition zu erhalten.
Es gibt viele verschiedene Arten um eine Clusterung vorzunehmen. Eine der Bekanntesten wäre zum Beispiel k–means (s. Jancey [12]), der euklidische Daten benötigt.
Das spektrale Clustern hingegen verwendet nur einen ungerichteten, positiv gewichteten Graphen, was einen Vorteil darstellt. Denn es gibt die Möglichkeit einer genaueren Vorverarbeitung (Transformation) der Merkmale. Geht man von einem metrischen Raum aus, so kann man zum Beispiel direkt aus den Merkmalen einen ungerichteten, positiv gewichteten Graphen erzeugen. Oder man verwendet eine Metrik (z.B. euklidischen Abstand) und definiert damit eine Unähnlichkeit, wandelt diese in eine Ähnlichkeit, und zeichnet einen vollständigen ungerichteten, positiv gewichteten Graphen (hat maximale Kantenanzahl) mit den Datenpunkten als Knoten und den Ähnlichkeiten als Kantengewichte. Aus diesem so gewonnen ungerichteten, positiv gewichteten Graphen könnte

man einen weiteren ungerichteten, positiv gewichteten Graphen generieren, indem man zum Beispiel alle Kanten entfernt, die kleiner als ein festgelegter Wert sind (ϵ–Graph s. Luxburg [13]).

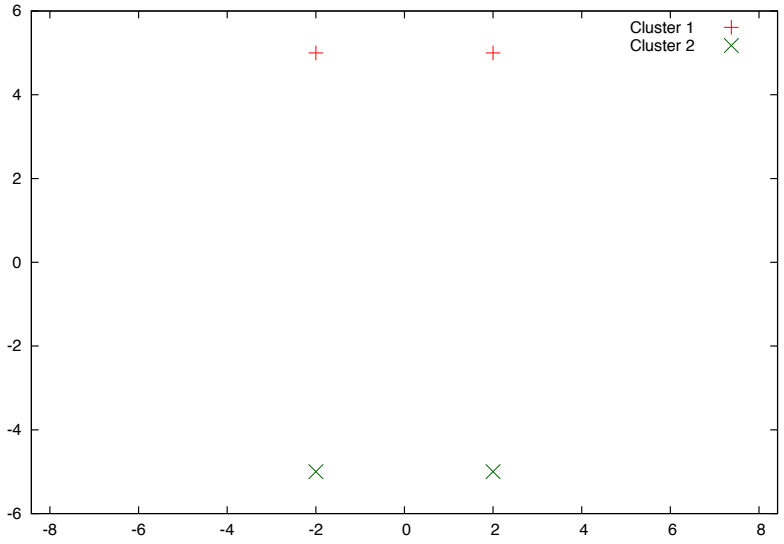

Abbildung 2.4.: y-Merkmal dominiert x-Merkmal.

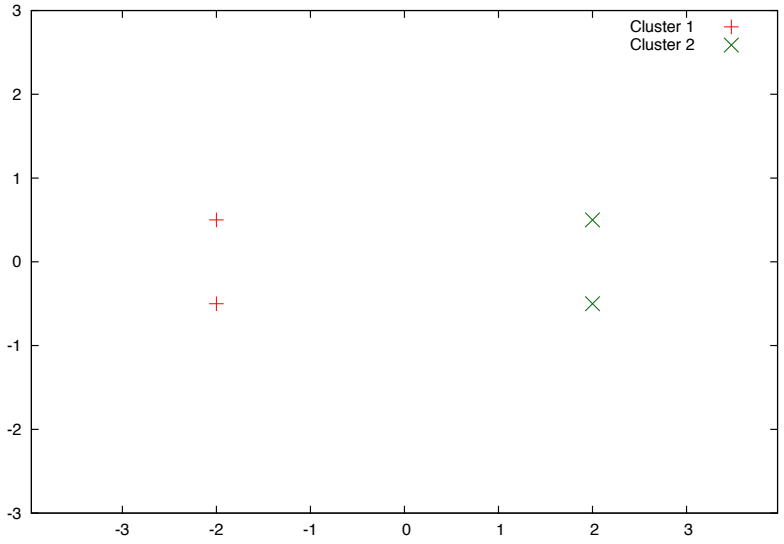

Abbildung 2.5.: x–Merkmal dominiert y–Merkmal.

2.4. Unser Vorgehen bei Microarray–Daten

Wie in dem zuletzt genannten Beispiel werden die betrachteten Merkmale schrittweise in einen ungerichteten, positiv gewichteten Graphen umgewandelt. Bei den Betrachtungen liegen die Merkmale in dem R^m vor, weshalb als Metrik der euklidische Abstand verwendet wird und dadurch eine Unähnlichkeit erhalten wird. Da man von endlich vielen Datenpunkten ausgeht, gibt es einen maximalen Abstand (Unähnlichkeit) s_0 zwischen zwei Datenpunkten. Sei $d_{i,j}$ der euklidische Abstand von Punkt i zu Punkt j, so ist durch $s(i,j) = s_0 - d_{i,j}$ eine Ähnlichkeit definiert. Daraus erhält man einen vollständig ungerichteten, positiv gewichteten Graphen und löscht aus diesem alle Kanten (i,i) für alle Knoten i. In den Abbildungen 2.6, 2.7, 2.8 und 2.9 kann dieses Vorgehen schrittweise betrachtet werden.

Abbildung 2.6.: Beispiel von drei Datenpunkten in $\mathbb{R}$

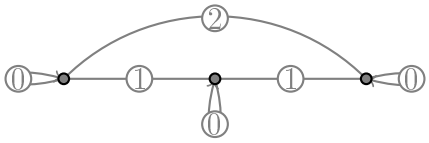

Abbildung 2.7.: Beispiel einer Unähnlichkeit als ungerichteter, positiv gewichteter Graph dargestellt (aus Abb. 2.6 gewonnen)

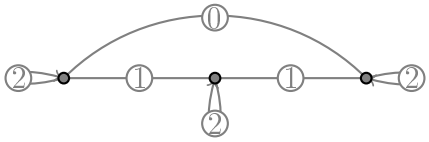

Abbildung 2.8.: Beispiel einer Ähnlichkeit als ungerichteter, positiv gewichteter Graph dargestellt (aus Abb. 2.7 gewonnen)

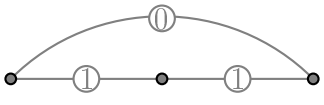

Abbildung 2.9.: Resultierender ungerichteter, positiv gewichteter Graph (aus Abb. 2.8 gewonnen)

2.5. Die Zusammenfassung

Die zentrale Aussage dieses Kapitels ist, dass sich die Semantik einer Partition hinter den Ähnlichkeiten verbirgt. Ist die Vorverarbeitung (Merkmalstransformation) nicht gut geglückt, so wird kein Clusteralgorithmus zuverlässig verwertbare Resultate erbringen.

Kapitel 3.

Die Bewertung von Partitionen

Dieses Kapitel behandelt die Bewertung von Partitionen, um sie miteinander vergleichen zu können. Ist man dazu in der Lage, so muss man nur noch alle Möglichkeiten der Partitionen durchlaufen, was in der Realität jedoch sehr aufwendig ist. Denn bei n Datenpunkten gibt es $2^{n-1}-1$ Möglichkeiten die Datenpunkte auf zwei Cluster aufzuteilen, worauf später noch genauer eingegangen wird. Alleine durch das Durchlaufen aller Partitionen, hat man ein exponentielles Wachstum.
Aus diesem Grund wird oft eine Approximation angewandt, welche nicht in einem globalen Optimum enden muss. Um einen Clusteralgorithmus studieren zu können, wird zunächst ein schneller Algorithmus entwickeln, der uns das globale Optimum berechnet. Später wird die Güte der Approximation durch den Vergleich mit dem optimalen Ergebnis illustriert.

3.1. Die Clusterkriterien

Ein Clusterkriterium ist eine Abbildung, die eine Familie von disjunkten Mengen auf eine Menge mit totaler Ordung abbildet. In der Literatur werden oft weitere Einschränkungen getroffen, so ist die Vereinigung über die Familie immer endlich (endliche Anzahl von Datenpunkten), und wird auf die nichtnegativen reellen Zahlen abgebildet.

3.2. Wichtige Definitionen und Vorbereitungen

Zunächst benötigt man einige Definitionen um Clusterkriterien zu entwickeln.

3.2.1. Die Partition

Definition 3.1 *Gegeben sei ein ungerichteter, positiv gewichteter Graph $G = (V, E, f)$ wobei $V = \{v_1, \ldots, v_n\}$. Eine **Partition** von G ist eine Menge von Teilmengen von V ($C_i \subset V$ wobei $i \in \{1, \ldots, K\}$ und $K \leq n$ ist) mit den Eigenschaften:*

(i) $C_i \neq \emptyset, i = 1, \ldots, K$

(ii) $\bigcup_{i=1}^{K} C_i = V$

(iii) $C_i \cap C_j = \emptyset, i, j = 1, \ldots, K$ *und* $i \neq j$

3.2.2. Der cut

Spektrales Clustern arbeitet auf ungerichteten, positiv gewichteten Graphen. Die Idee des *cut* ist es einen ungerichteten, positiv gewichteten Graphen durchzuschneiden, so dass mehrere kleinere ungerichtete, positiv gewichtete Graphen entstehen. Diese so geschaffenen ungerichteten, positiv gewichteten Graphen entsprechen den Clustern. Datenpunkte aus verschiedenen Clustern sollten stets geringe Ähnlichkeit aufweisen. Der erste Ansatz hierzu ist die Summe der Kantengewichte, die zwischen zwei Clustern verlaufen, zu minimieren (Abb. 3.1).
Die Originaldefinition war nur auf unterschiedlichen Clustern Definiert (s. Luxburg [13]). Der $MinMaxCut$ benötigt jedoch den *cut* eines Clusters, weshalb die Bedingung $C_i \neq C_j$ weggelassen wurde (s. Ding et al. [5]).

Definition 3.2 *(s. Luxburg [13]) (etwas verallgemeinert) Sei $G = (V, E, f)$ ein ungerichteter, positiv gewichteter Graph und $(C_i)_{i \in \{1,\ldots,K\}}$ eine Partition. Man summiert die Kantengewichte, die zwischen den Clustern verlaufen, auf:*

$$\mathrm{cut}(C_i, C_j) = \sum_{(a,b) \in E,\ a \in C_i \text{ und } b \in C_j} f(a, b)$$

wobei $i, j \in \{1, \ldots, K\}$. Verallgemeinert gilt dann:

$$\mathrm{cut}(C_1, \ldots, C_K) = \sum_{i \in \{1,\ldots,K\}} \mathrm{cut}(C_i, \bar{C}_i)$$

wobei $\bar{C}_i = V \setminus C_i$.

Abbildung 3.1.: Summe roter Kanten entspricht dem cut(Cluster1, Cluster2). Kantengewichte in der Darstellung weggelassen.

3.2.3. Der Grad eines Knotens

Kanten (von ungerichteten, positiv gewichteten Graphen) sind zwei– oder einelementige Mengen. Der **Grad eines Knotens** ist die Summe aller Kanten, von denen er Element ist (Abb. 3.2).

Definition 3.3 *(s. Luxburg [13]) Sei $G = (V, E, f)$ ein ungerichteter, positiv gewichteter Graph. Der **Grad eines Knotens** i ist definiert als:*

$$d_i := \sum_{\{i,j\} \in E} f(i,j)$$

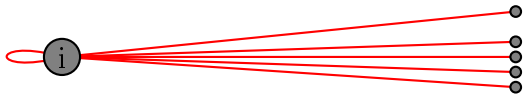

Abbildung 3.2.: Summe roter Kanten entspricht d_i. Kantengewichte in der Darstellung weggelassen.

3.2.4. Das Volumen eines Clusters

Das Volumen wird später benötigt, um den *cut* zu gewichten.

Definition 3.4 *(s. Luxburg [13]) Sei $G = (V, E, f)$ ein ungerichteter, positiv gewichteter Graph und $(C_i)_i$ $i \in \{1..K\}$ eine Partition. Das Volumen eines Clusters ist*

$$\mathrm{vol}(C_i) := \sum_{j \in C_i} d_j$$

3.2.5. Der zusammenhängende Graph

Man sagt ein ungerichteter, positiv gewichteter Graph ist **unzusammenhängend**, wenn es eine Partition gibt, so dass zwischen den Clustern keine Kante verläuft (Abb. 3.3). Er ist **zusammenhängend**, wenn jede Partition Kanten zwischen den Clustern enthält.

Definition 3.5 *Man nennt einen ungerichteten, positiv gewichteten Graphen* ***zusammenhängend****, falls es keine Partition* (C_1, C_2) *gibt mit* $\text{cut}(C_1, C_2) = 0$.

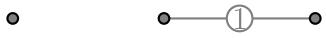

Abbildung 3.3.: Unzusammenhängender Graph

3.3. Die Ideen der Clusterkriterien

Zunächst geht man von zwei Clustern aus, um die Ideen der Clusterkriterien zu erläutern.
Die erste Idee eines Clusterkriteriums ist, dass Cluster untereinander sehr verschieden sein sollen (die Ähnlichkeit soll zwischen den Clustern minimal sein). Anders ausgedrückt, der $cut(C_1, C_2)$ zwischen den zwei Clustern C_1 und C_2 soll minimal sein, also am weitesten auseinander liegen. Diese erste Idee führt jedoch oft zu sehr unbalancierten Clustern (die Cluster haben sehr unterschiedlich viele Datenpunkte inne), weshalb diese Idee alleine nicht genügt. Um eine Balanciertheit zu erreichen, könnte man den $cut(C_1, C_2)$ in Verhältnis zur minimalen Anzahl der Datenpunkte innerhalb eines Clusters setzen. Man gewichtet die Schnitte, was eine gewisse Balanciertheit garantiert (s. Kannan et al. [18]). *RatioCut* (s. Hagen und Kahng [10]) und *NormalCut* (s. Shi und Malik [24]), die gleich betrachtet werden, sind sehr ähnlich aufgebaut.
Ein anderer Zugang wäre Folgender: Betrachtet man die Funktion $g_{c_i} := \frac{c_i}{a_i} + \frac{c_i}{b_i}$ (Beispiel in Abb. 3.4) mit $a_i + b_i = konst$ und $c_i \in [0, \infty[, a_i, b_i \in [0, konst]$, so findet man ihr Minimum bei $a_i = b_i$. Seien nun a_i, b_i nicht mehr unabhängig von c_i, das heißt, wählt man ein c_i, so wählt man dadurch auch a_i und b_i (bei dem *RatioCut* entspricht c_i dem $cut(C_{1_i}, C_{2_i})$ einer zwei Cluster Partition $C_i = (C_{1_i}, C_{2_i})$, und a_i, b_i entspricht der Anzahl der Datenpunkte innerhalb der Cluster). Weiterhin sei c_0, und damit auch a_0 und b_0 fixiert, wobei a_0 viel größer ist als b_0 (betrachte eine unbalancierte Partition C_0). So könnte man zum Beispiel ein c_1 finden, das nicht viel größer als c_0 ist **und die Eigenschaft** $a_1 = b_1$ hat, wodurch $g_{c_1} < g_{c_0}$ gilt (betrachte eine Partition C_1 deren $cut(C_{1_1}, C_{2_1}) > cut(C_{1_0}, C_{2_0})$ ist, aber balancierte Cluster enthält). Durch die Gewichtung der Summanden wird erreicht, dass der Clusterkriteriumswert $g_{c_1} < g_{c_0}$

ist, obwohl der $cut(C_{1_1}, C_{2_1}) > cut(C_{1_0}, C_{2_0})$ ist. Um also besser balancierte Cluster zu erhalten, minimiert man nicht über den $cut(C_{1_i}, C_{2_i})$ sondern über g_{c_i}. Dieser erste Verbesserungsansatz entspricht dem *RatioCut*.

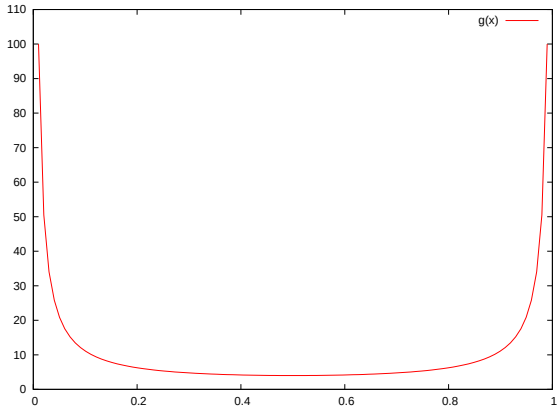

Abbildung 3.4.: $g(x) = g_{c_i}$ mit $c_i = 1, a_i = x$ und somit $b_i = 1 - x$

Bei dem *NormalCut* ist die Vorgehensweise äquivalent, nur verwendet man hier statt der Anzahl der Datenpunkte, das Volumen ($vol(C_1)$ und $vol(C_2)$). Hierbei ist zu beachten, dass bei jeder Partition $C_i = (C_{1_i}, C_{2_i})$ das Volumen $vol(C_{1_i}) + vol(C_{2_i}) = vol(V)$ konstant ist.
Eine weitere Idee eine Partition zu finden wird über das Merkmal der Kompaktheit entwickelt, weshalb alle Datenpunkte innerhalb eines Clusters sehr ähnlich sein sollten. Diese Kompaktheit eines Clusters C_1 kann man wieder mit dem $cut(C_1, C_1)$ berechnen. Hier werden alle Kanten innerhalb eines Clusters zweimal aufsummiert und entsprechen somit dem physikalischen Volumen bis auf ein Vielfaches. Tatsächlich haben $cut(C_1, C_1)$ und $vol(C_1)$ etwas gemeinsam. So gilt $vol(C_1) = cut(C_1, C_2) + cut(C_1, C_1)$. Der *MinMaxCut* (s. Ding et al. [6]) verwendet die Idee der Verschiedenheit von Clustern und deren Kompaktheit, und ist eine Abwandlung des *NormalCut*.

3.3.1. Das Vorgehen bei unseren Beispielen

Die verwendeten Clusterbeispiele in diesem Kapitel wurden wie folgt generiert. Es liegt immer der metrische Raum $\mathbb{R}^2$ mit dem euklidischen Abstandsmaß zugrunde. Aus dem Abstandsmaß werden die Unähnlichkeiten und somit sowohl die Ähnlichkeiten, als auch anschließend der ungerichtete, positiv gewichtete Graph, ohne selbstreferenzierende Kanten erzeugt. Aus diesem Graph werden alle möglichen Partitionen durchlaufen und durch ein Clusterkriterium bewertet. Leider sind die Clusterkriterien nicht injektiv, weshalb man nicht von der optimalen Partition, sondern nur von einer optimalen Partition sprechen kann. Hierbei ist eine optimale Partition, eine Partition, welche bezüglich eines Clusterkriteriums den minimalen Wert annimmt.

3.4. Der RatioCut

Wie bereits erwähnt, verwendet der *RatioCut* eine Gewichtung des $cut(C_i, \bar{C}_i)$, um gleich große Cluster zu erhalten (Anzahl der Datenpunkte innerhalb der Cluster sind gleich). Man hofft darauf, dass durch eine kleine Verschlechterung des $cut(C_i, \bar{C}_i)$ ungefähr gleich große Cluster entstehen, und dabei die Summe kleiner wird.

Definition 3.6 *(s. Luxburg [13])*

$$\text{RatioCut}(C_1, ..., C_n) = \sum_{i \in \{1..n\}} \frac{\text{cut}(C_i, \bar{C}_i)}{|C_i|}$$

wobei $|C_i|$ die Anzahl der Datenpunkte in C_i ist.

Tatsächlich ist der *RatioCut* in Spezialfällen in der Lage gleich große Cluster zu selektieren, wie man in Abbildung 3.5 sehen kann. Dies ist jedoch nicht immer der Fall. Betrachtet man die Abbildung 3.6, so würde man eine andere Partition bevorzugen.

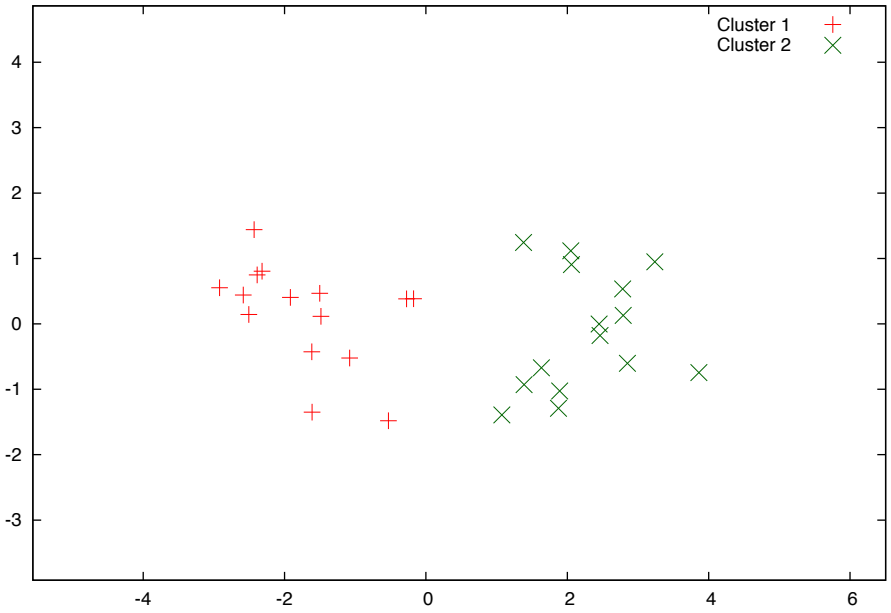

Abbildung 3.5.: Richtiges clustern mit Hilfe des RatioCut.

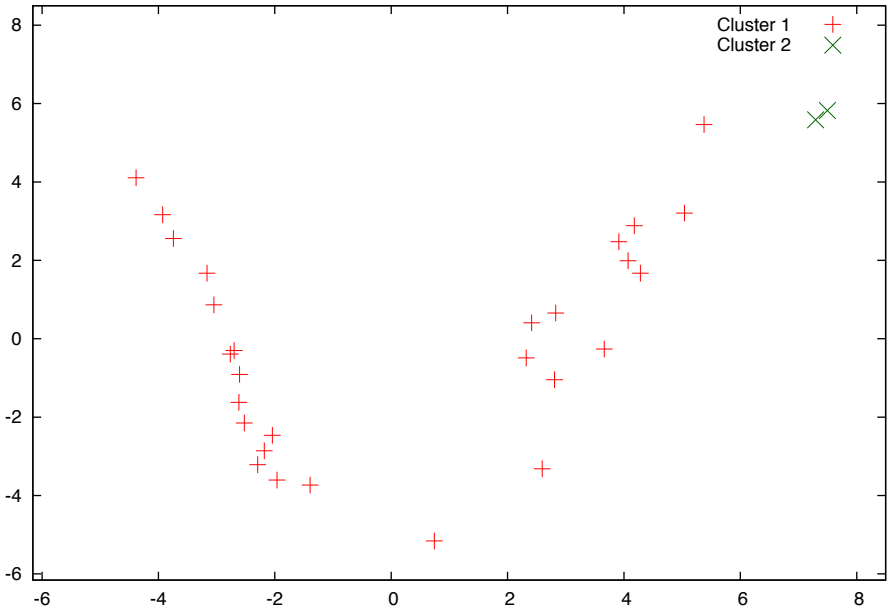

Abbildung 3.6.: Schlechtes clustern mit Hilfe des RatioCut.

3.5. Der NormalCut

Der *NormalCut* unterscheidet sich vom *RatioCut* nur durch seine Nenner, welche den jeweiligen Volumina entsprechen. Die Intention ist hier die selbe, nur das Maß für die Größe eines Clusters ist anders.

Definition 3.7 *(s. Luxburg [13])*

$$\text{NormalCut}(C_1, ..., C_n) = \sum_{i \in \{1..n\}} \frac{\text{cut}(C_i, \bar{C}_i)}{\text{vol}(C_i)}$$

Wie bereits erwähnt, besteht die Verbesserung darin, statt der Kardinalität eines Clusters sein Volumen *vol* zu verwenden, wodurch meist bessere Partitionen entstehen (vergleiche Abb. 3.7 und Abb. 3.8).

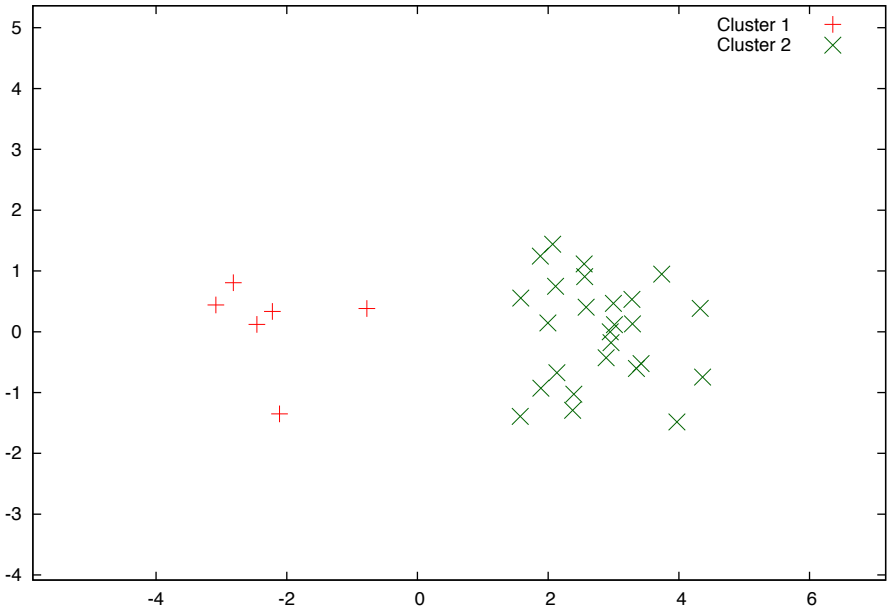

Abbildung 3.7.: Richtiges clustern dank NormalCut.

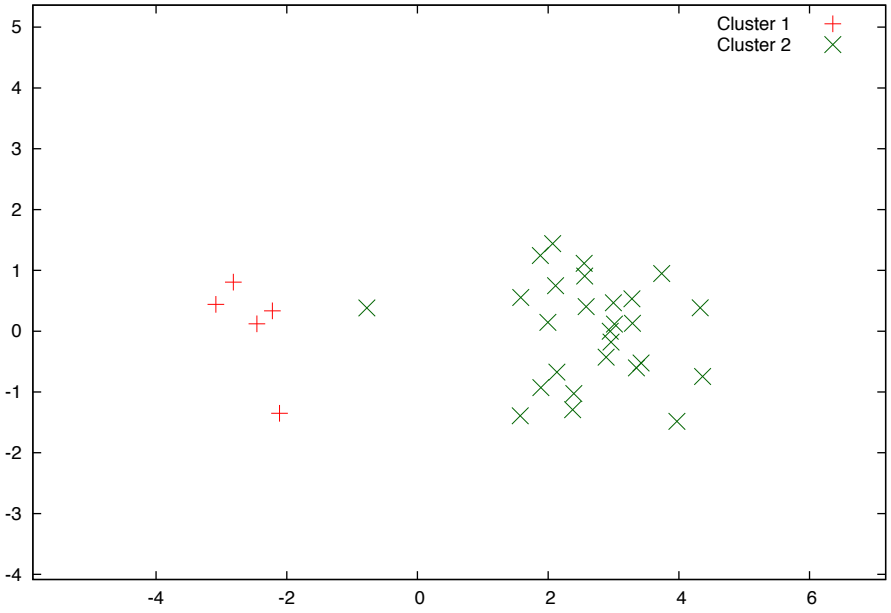

Abbildung 3.8.: Ein Punkt ist falsch geclustert von RatioCut.

3.6. Der MinMaxCut

Im Gegensatz zu *RatioCut* und *NormalCut* hat der *MinMaxCut* noch eine zweite Motivation: Er versucht die Kompaktheit eines Clusters mit einzubeziehen, indem er den *cut* zwischen den Clustern minimiert und die Kompaktheit (*cut* innerhalb eines Clusters) maximiert.

Definition 3.8 *(s. He et al. [9])*

$$\text{MinMaxCut}(C_1, ..., C_n) = \sum_{i \in \{1..n\}} \frac{\text{cut}(C_i, \bar{C}_i)}{\text{cut}(C_i, C_i)}$$

MinMaxCut tendiert zu gleich großen Clustern wie man in Abbildung 3.9 und Abbildung 3.10 sehen kann.

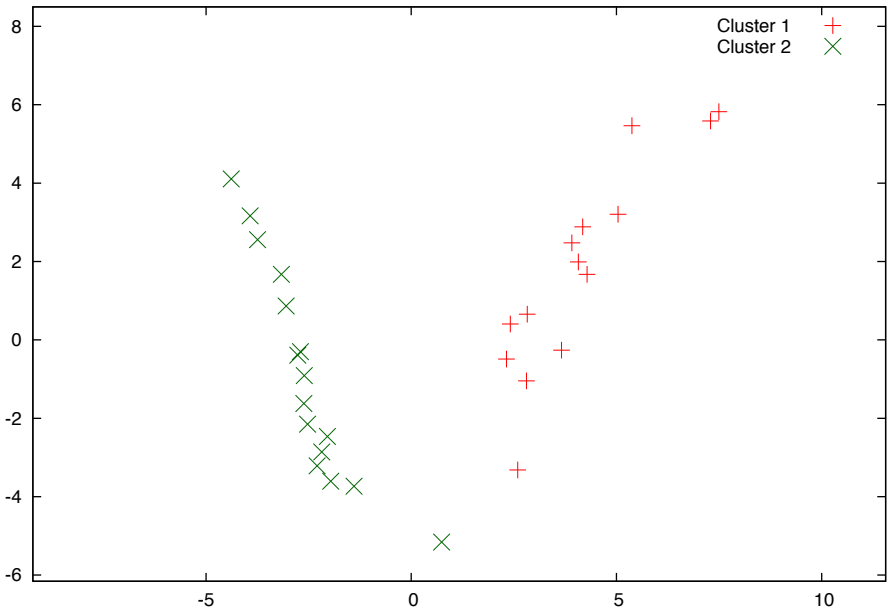

Abbildung 3.9.: Bessere, aber nicht perfekte Partition durch *MinMaxCut* (quadratischer euklidischer Abstand wurde verwendet).

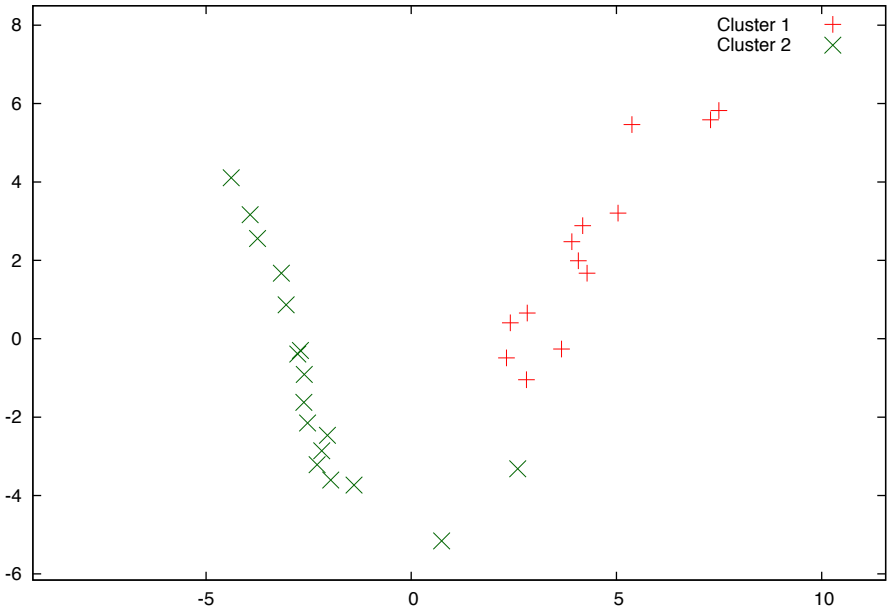

Abbildung 3.10.: *NormalCut* Clustert zwei Datenpunkte falsch (quadratischer euklidischer Abstand wurde verwendet).

3.7. Die Cluster Balance

Wie gerade erwähnt, tendiert *MinMaxCut* zu gleich großen Clustern. In Ding et al. [6] findet man eine interessante Betrachtung eines ungerichteten, positiv gewichteten Graphen. Dort wird ein vollständiger ungerichteter, positiv gewichteter Graph $G = (\{1, ..., 2 * n\}, \binom{2*n}{2}, f)$ betrachtet (ohne selbstreferenzierende Kanten), wobei $f(\{i, j\}) = 1$ eine konstante Funktion ist und $n \in \mathbb{N}$ ist.
Will man nun diesen Graphen in zwei Cluster (C_1 und C_2) aufspalten, so kann man Folgendes beobachten:

$$\text{cut}(C_1, C_2) = |C_1| * |C_2|,$$

da ein Knoten in C_1 $|C_2|$ Kanten hat, die in C_2 landen und es in C_1 $|C_1|$ Knoten gibt. Für das Volumen eines Clusters C_1 gilt

$$\text{vol}(C_1) = |C_1|(2n - 1),$$

da ein Knoten $2n - 1$ Kanten hat und es $|C_1|$ Knoten in C_1 gibt. Für den *RatioCut* folgt:

$$\text{RatioCut}(C_1, C_2) = \frac{\text{cut}(C_1, C_2)}{|C_1|} + \frac{\text{cut}(C_1, C_2)}{|C_2|} = \frac{|C_1||C_2|}{|C_1|} + \frac{|C_1||C_2|}{|C_2|} = \\ = (|C_1| + |C_2|) = 2n.$$

Für den *NormalCut* folgt:

$$\text{NormalCut}(C_1, C_2) = \frac{\text{cut}(C_1, C_2)}{\text{vol}(C_1)} + \frac{\text{cut}(C_1, C_2)}{\text{vol}(C_2)} = \\ = \frac{|C_1||C_2|}{|C_1|(2n-1)} + \frac{|C_1||C_2|}{|C_2|(2n-1)} = \frac{2n}{2n-1}.$$

Für den *MinMaxCut* folgt:

$$\text{MinMaxCut}(C_1, C_2) = \frac{\text{cut}(C_1, C_2)}{\text{cut}(C_1, C_1)} + \frac{\text{cut}(C_1, C_2)}{\text{cut}(C_2, C_2)} = \\ = \frac{|C_1||C_2|}{|C_1| * (|C_1| - 1)} + \frac{|C_1||C_2|}{|C_2| * (|C_2| - 1)} = \frac{|C_2|}{|C_1| - 1} + \frac{|C_1|}{|C_2| - 1}.$$

Festzustellen ist, dass die Ergebnisse von *RatioCut* und *NormalCut* konstant sind. Sie hängen nicht von einer gewählten Partition ab. Somit gibt es hier keinerlei Beschränkungen wie die Cluster auszusehen haben.
Hingegen ist das Ergebnis von *cut* abhängig von einer gewählten Partition und hat als Minimum $1 * (2 * n - 1)$, womit es als Clusterkriterium einen Knoten isolieren würde.
Betrachtet man das Ergebnis von *MinMaxCut*, so hängt dieses, wie *cut*, auch von einer Partition ab und hat als minimalen Wert $\frac{2n}{n-1}$, welcher nur bei $|C_1| = |C_2| = n$ erreicht wird. Der *MinMaxCut* würde also gleich große Cluster erzeugen.

3.8. Die Effektivität bei nahezu optimalen Ähnlichkeiten

Bisher wurden in diesem Kapitel Clusterkriterien eingeführt, wurden auf die Eigenschaften der jeweiligen Clusterkriterien eingegangen und wurde festgestellt wie man optimale Partitionen berechnet.
Wie schon erwähnt hängt die Güte einer Partition stark von seiner Merkmalstransformation ab, da sie die optimale Partition entscheidend beeinflusst. Eine perfekte Transformation wäre, wenn alle Datenpunkte innerhalb eines Clusters eine Ähnlichkeit größer 0 hätten und Datenpunkte zwischen Clustern keine Ähnlichkeit aufweisen würden. Bei

zwei Clustern wäre eine Ähnlichkeitsmatrix (an der Stelle i, j steht die Ähnlichkeit zwischen Punkt i und j) der Art $\begin{Bmatrix} C_0 & 0 \\ 0 & C_1 \end{Bmatrix}$ optimal, wobei C_0 und C_1 Ähnlichkeitsmatrizen sind, und alle Einträge größer 0 sind.
Hat man eine solche Ähnlichkeit gefunden, ist es trivial die „richtige“ Partition zu berechnen, da man nur nach Zusammenhangskomponenten zu suchen braucht. Bei den von uns betrachteten Clusterkriterien wird der $cut(C_0, C_1) = 0$ und somit auch jeder Clusterkriteriumswert gleich 0. Da 0 der kleinste Wert ist der angenommen werden kann, wird sichergestellt, dass in diesem Fall richtig geclustert wurde.
Deshalb wird nun ein interessanterer Fall mit zwei Clustern betrachtet, bei dem die Ähnlichkeit von zwei Datenpunkten innerhalb eines Clusters gleich 1 und sonst $0,5$ entspricht. Anders als in Abbildung 3.11 dargestellt, betrachten wir jedoch einmal einen Fall mit $|C_0| = |C_1| = 15$, und einmal einen Fall mit $|C_0| = 22, |C_1| = 8$. Des weiteren lassen wir, wie in Abbildung 3.11 zu sehen ist, die Kanten die von einem Knoten in sich selbst münden (selbstreferenzierende Kanten) weg.

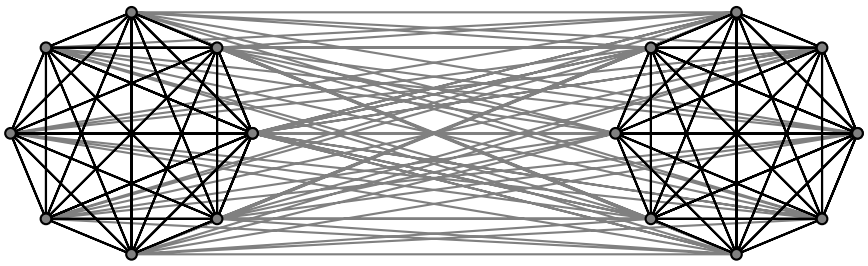

Abbildung 3.11.: Ungerichteter, positiv gewichteter Graph mit jeweils acht Datenpunkten innerhalb eines Clusters. Die schwarzen Kanten entsprechen einer Ähnlichkeit von 1, während die grauen einer Ähnlichkeit von $0,5$ entsprechen.

Alle betrachteten Clusterkriterien (*RatioCut*, *NomalCut* und $MinMaxCut$) lösen diese Aufgabe perfekt. Keiner von ihnen clustert auch nur einen Datenpunkt falsch. Die Kriteriumswerte können ähnlich zu Abschnitt 3.7 berechnet und in den Tabellen 3.1 und 3.2 eingesehen werden.
Betrachtet man jedoch einen entarteten Fall mit $|C_0| = 26, |C_1| = 4$, so clustert MinMaxCut sehr falsch und ordnet beiden Clustern fünfzehn Datenpunkte zu (Tabelle 3.3). Es werden also elf Datenpunkte falsch zugewiesen. Die beiden anderen Clusterkriterien lösen selbst diesen entarteten Fall perfekt.
Alle in diesem Kapitel optimalen Partitionen wurden mit Hilfe eines optimierten exhaustiven Algorithmus berechnet, der im folgenden Kapitel erläutert wird (bestimmt Clusterkriteriumswerte aller möglichen Partitionen und davon die Minimalstelle).

Kriterium	$\|C_0\| = \|C_1\| = 15$	ein Punkt falsch zugeordnet (e.P.f.z.) $\|C_0\| = 16, \|C_1\| = 14$
$\text{cut}(C_0, C_1)$	$112,5$	119
$\text{cut}(C_0, C_0)$	210	225
$\text{cut}(C_1, C_1)$	210	182
$\text{vol}(C_0)$	$337,5$	360
$\text{vol}(C_1)$	$337,5$	315
RatioCut	15	$\approx 15,94$
NormalCut	$\approx 0,67$	$\approx 0,71$
MinMaxCut	$\approx 1,07$	$\approx 1,18$

Tabelle 3.1.: Überblick aller Kriteriumswerte für $|C_0| = |C_1| = 15$

Kriterium	$\|C_0\| = 22, \|C_1\| = 8$	e.P.f.z. $\|C_0\| = 21, \|C_1\| = 9$	e.P.f.z. $\|C_0\| = 23, \|C_1\| = 7$
$\text{cut}(C_0, C_1)$	88	105	84
$\text{cut}(C_0, C_0)$	462	420	484
$\text{cut}(C_1, C_1)$	56	64	42
$\text{vol}(C_0)$	572	546	591
$\text{vol}(C_1)$	152	178	133
RatioCut	15	$\approx 16,67$	$\approx 15,65$
NormalCut	$\approx 0,73$	$\approx 0,78$	$\approx 0,77$
MinMaxCut	$\approx 1,76$	$\approx 1,89$	$\approx 2,17$

Tabelle 3.2.: Überblick aller Kriteriumswerte für $|C_0| = 22, |C_1| = 8$

Kriterium	$\|C_0\| = 26, \|C_1\| = 4$	e.P.f.z. $\|C_0\| = 25, \|C_1\| = 5$	e.P.f.z. $\|C_0\| = 27, \|C_1\| = 3$	elf P.f.z. $\|C_0\| = \|C_1\|$
$\text{cut}(C_0, C_1)$	52	75	42	195
$\text{cut}(C_0, C_0)$	650	600	676	166
$\text{cut}(C_1, C_1)$	12	16	6	210
$\text{vol}(C_0)$	728	700	745	405
$\text{vol}(C_1)$	68	96	51	361
RatioCut	15	18	$\approx 15,56$	26
NormalCut	$\approx 0,84$	$\approx 0,89$	$\approx 0,88$	$\approx 1,02$
MinMaxCut	$\approx 4,41$	$\approx 4,81$	$\approx 7,06$	$\approx 2,1$

Tabelle 3.3.: Überblick aller Kriteriumswerte für $|C_0| = 26, |C_1| = 4$

3.9. Zusammenfassung der Kriterien

Allgemein kann man sagen, dass *NormalCut* bessere Resultate als *RatioCut* liefert. Scheinbar ist das *vol* ein besseres Maß für die Größe eines Clusters, als die Kardinalität. Bei Testläufen in Kombination aus Datensätzen, Ähnlichkeiten und dem *RatioCut* beziehungsweise *NormalCut* wird diese Aussage untermauert.
Das Verhältnis von *NormalCut* und *MinMaxCut* lässt sich jedoch nicht so leicht pauschalisieren. Als erstes ist festzustellen, dass beide oft ähnlich gute Ergebnisse liefern. Das mag daran liegen, dass sich bei einem kleinen *cut*, die beiden unterschiedlichen Nenner der Clusterkriterien nur noch marginal unterscheiden. Weiß man vorab, dass es sich um zwei gleich große Cluster handelt, so sollte man *MinMaxCut* priorisieren. Den *NormalCut* sollte man verwenden, wenn man keinerlei Informationen bezüglich der Clustergrößen hat. Es handelt sich hierbei jedoch um eine Verallgemeinerung und kann in Spezialfällen sicher auch falsch sein.
Hat man nach der Merkmalstransformation eine Ähnlichkeit gefunden, so stellt sich die Frage, ob man diese Transformation weiterführen soll und somit die Ähnlichkeit in einen ungerichteten, positiv gewichteten Graphen überführt. Dies kann durchaus notwendig und von Bedeutung sein. Setzt man die Diagonale der Ähnlichkeitsmatrix auf 0, so lässt man graphentheoretisch nur die selbstreferenzierenden Kanten weg. Bei dem *RatioCut* hat diese Transformation keinerlei Bedeutung, da man nur Kanten zwischen den Clustern betrachtet. Bei den anderen beiden Clusterkriterien könnte es durchaus zu verschiedenen Effekten kommen.
Laut Luxburg [13] liefern *RatioCut* und *NormalCut* bei regulären ungerichteten, positiv gewichteten Graphen (jeder Knoten hat denselben Grad) ähnliche Partitionen. Als Beispiel hierzu kann Abschnitt 3.7 herangezogen werden.

Kapitel 4.

Allgemeines Problem beim Clustern und seine Optimierung

Wir wissen nun was Clusterkriterien sind und wie man eine Partition bewertet. Was noch fehlt ist ein Algorithmus, welcher eine Partition mit minimalem Wert bezüglich eines Clusterkriteriums berechnet. Da die Anzahl der zu clusternden Daten immer endlich ist, gibt es auch nur endlich viele Arten Partitionen zu wählen. Bei manchen Clusterkriterien gibt es jedoch Möglichkeiten, diesen exhaustiven Ansatz zu verfeinern. So kann man manchmal, wie zum Beispiel bei sinlge Linkage (s. Ritter [21]), sehr viel Rechenzeit einsparen. Leider gehören die hier vorgestellten Clusterkriterien nicht dazu, denn sie sind *NP–hart* (s. Luxburg [13]) und somit sehr schwer zu berechnen. Um trotzdem die Clusterkriterien zu studieren, war die Entwicklung eines optimierten exhaustiven Ansatzes existenziell. Die Idee hinter diesem Ansatz ist, statt das Clusterkriterium immer wieder neu zu berechnen, es nur zu aktualisieren.
Da es hierbei nur um das Studium der Clusterkriterien geht, spezialisiere ich mich jetzt auf den 2–Clusterfall. Dafür werde ich den von mir entwickelten $FRTPS$–Algorithmus (Fast Run Through Power–Set) einführen. Dieser generiert die Menge aller Bitvektoren auf eine sehr spezielle Art und Weise. Für uns ist der Rückgabewert nicht interessant, dafür aber wie der Algorithmus diesen berechnet. Bei einem n–stelligen Bitvektor, gibt es bekannterweise 2^n Elemente (oder auch Zahlen) die damit darstellbar sind. Es werden jedoch nur die ersten 2^{n-1} Elemente benötigt, weshalb der $FRTPS$–Algorithmus nur zur Hälfte laufen muss.

4.1. Die Potenzmenge und Partition

Nun zu dem Spezialfall einer Clusterung mit zwei Clustern ($Cluster1$ und $Cluster2$). Um das Optimum eines Clusterkriteriums zu bestimmen, muss man jede mögliche Kombination einer Partition in Erwägung ziehen. Betrachten man zunächst $Cluster1$, so ist

der erste Ansatz, dass jede Teilmenge der Datenpunkte einmal in diesem Cluster enthalten sein muss. Hier kommt nun die Potenzmenge ins Spiel, welche alle Möglichkeiten, wie $Cluster1$ zusammengesetzt sein könnte, beinhaltet. Indem man $Cluster1$ eine Zuordnung gibt, impliziert dies auch eine Zuordnung für $Cluster2$. Die Zuordnung von $Cluster2$ braucht man für $Cluster1$ nicht mehr zu betrachten, da nicht die Zuordnung an sich von Interesse ist, sondern lediglich die Zerlegung der Datenpunkte.
Da die Komplementbildung *injektiv* ist, hat jedes Element in der Potenzmenge sein eindeutiges Komplement. Betrachtet man nun diese Anzahl von Zerlegungen der Datenpunkte, so kann man feststellen, dass diese der Hälfte der Kardinalität der Potenzmenge entspricht, also $2^n/2 = 2^{n-1}$ Elemente beinhaltet. Da ein Cluster niemals leer sein darf (leere Menge ist Element der Potenzmenge), muss man noch 1 abziehen und kommt so auf die Anzahl aller möglichen Zerlegungen.
Wir wissen jetzt über die Komplexität des Problems bescheid und haben einen ersten Ansatz gefunden, alle Möglichkeiten zu durchlaufen. Es gibt jedoch noch viele weitere Probleme die zu lösen sind.

4.2. Die partielle Ordnung der Potenzmenge

Das erste Problem ist es die Potenzmenge effektiv zu berechnen. Angenommen man hat dieses Problem gelöst, so ist es erstrebenswert, diese Menge partiell zu ordnen (Baumstruktur)(*)[1], so dass der Nachfolger einer Menge (Kind) sich immer nur um ein Element unterscheidet. Ist eine solche Ordnung gefunden, so kann man anstatt das Clusterkriterium immer wieder neu zu berechnen, eine Aktualisierung des Vorgängers durchführen und so Berechnungszeit sparen. Wie bereits gezeigt wurde, muss man nicht die ganze, sondern nur die halbe Potenzmenge durchlaufen. Dies kann man durch einen einfachen Mechanismus erreichen. Da die überflüssigen Mengen, die Komplemente der abgearbeiteten Mengen sind, gleicht man das Komplement jeder neuen Menge mit den bereits abgearbeiteten Mengen ab. Dadurch entsteht aber ein großes Problem: Der Speicherverbrauch steigt exponentiell an, was rasch zu einem uneffizienten Programm führt, da der Arbeitsspeicher zu schnell gefüllt wird. Man muss neben der partiellen Ordnung (*) auch die Ordnung der Komplemente (**)[2] mit ins Spiel bringen. Ordnet man die Potenzmenge und nimmt man eine Menge aus der ersten Hälfte heraus, so muss das Komplement in der zweiten Hälfte liegen. Dadurch wird kein Wissen über die bereits abgearbeitete Menge benötigt und ein Abbruchkriterium durch ein schlichtes mitzählen kann eingeführt werden.
Zusammengefasst wird ein Algorithmus gesucht, der den Baum (partielle Ordnung) (*) durchläuft und dabei die Komplemente der bereits durchlaufenen Menge, erst nach 2^{n-1} erreicht.

[1] (*) ist eine Markierung für diese partielle Ordnung
[2] (**) ist eine Markierung für die Ordnung der Komplemente

4.3. Der Bitvektor und die Potenzmenge

Die Kardinalität einer Potenzmenge beträgt 2^n, wenn die Grundmenge n Elemente beinhaltet. Nimmt man einen n–stelligen Bitvektor, so sind damit 2^n Zahlen darstellbar. Die erste Idee ist es also, eine *Bijektion* zu finden, die jeden Bitvektor (also eine natürliche Zahl zwischen 0 und $2^n - 1$) auf ein Element der Potenzmenge abbildet. Dazu braucht man zunächst eine totale Ordnung der Grundmenge, um diese zu indizieren. Nimmt man nun eine beliebige Teilmenge, so kann man diese auch durch die Indizes darstellen. An den gleichen Indizes des Bitvektors schreibt man eine 1, ansonsten eine 0, womit eine Darstellung durch einen Bitvektoren gefunden wurde.
Formal gilt:

Satz 4.1 *Sei M eine beliebig endliche Menge mit Kardinalität n,* Pot *die Potenzmengenabbildung, g eine beliebige Bijektion von M auf $\{0, \ldots, n-1\}$ (eine Indexabbildung) und B die Menge aller möglichen n–stelligen Bitvektoren. Ist $f : \mathrm{Pot}(M) \to B$ definiert durch:*

Sei $a \in \mathrm{Pot}(M)$ $a \neq \emptyset$, $a_i \in a$ und $b \in B$ mit $b_{g(a_i)} = 1$ sonst 0. Dann ist $f(a) = b$ und $f(\emptyset) = 0$.

dann ist f eine Bijektion.

Beispiel:
Betrachtet man die Menge $M = \{a, b, c, d\}$ mit $g(a) = 0, g(b) = 1, g(c) = 2, g(d) = 3$, so entspricht $f(\{b, c\}) = (0, 1, 1, 0)$.

4.4. Der optimale Durchlauf durch einen Bitvektoren

Um die Lesbarkeit der Bitvektoren zu verbessern, wird ab sofort auf Klammern und Kommata verzichtet. Eine Darstellung der Potenzmenge wurde gefunden, mit deren Hilfe man in der Lage ist, diese zu durchlaufen. Jetzt entspricht jeder dieser Bitvektoren einer natürlichen Zahl (2–adische Entwicklung). Die Menge aller Bitvektoren könnte man auf gewohnte Art durchlaufen, was jedoch nicht vorteilhaft wäre, da diese der geforderten Ordnung (*) nicht entspricht. So wäre zum Beispiel der Nachfolger von 0111, 1000 welcher sich nicht an einer, sondern an vier Stellen von seinem Vorgänger unterscheidet. Ich werde jetzt den am Anfang erwähnten, von mir entwickelten $FRTPS$–Algorithmus vorstellen, der alle bisher angeführten Ordnungskriterien erfüllt. Später wird ein geometrischer Zugang entwickelt. Die wichtigste Operation dieses Algorithmus ist die $shift$–Operation. Sie unterscheidet sich von der bekannten Computeroperation *shift–right* (in

C++: >>), hat jedoch Parallelen hierzu. Anstatt einen gesamten Vektor zu shiften, wie es der Computer machen würde, wird in unserem Fall lediglich ein Bit geshiftet. Dieses Bit ist immer die am weitesten links stehende 1 und wird um eine Stelle nach rechts verschoben, falls dort eine 0 steht.

Beispiel:
Betrachtet man 01001, dann ist $shift(01001) = 00101$ und $shift(shift(01001)) = 00011$.
Weiterhin gilt $shift(00001) = 00001$ (die Anzahl der Einsen bleibt immer gleich).

Des weiteren gibt es eine $newShift$–Operation die eine $shift$–Operation einleitet. Diese Operation setzt an Position Eins immer eine 1.

Beispiel:
Betrachtet man wieder 01001, so ist $newShift(01001) = 11001$.

Der $FRTPS(a)$ Algorithmus ist in Algorithm 1 dargestellt und in *Pseudocode* verfasst. Man kann die erzeugten Bitvektoren des Algorithmus als Baum darstellen, indem $newShift$ eine neue Ebene generiert, und $shift$ die Brüder. Da $shift$ die Anzahl der Einsen nicht verändert und $shift$ nur solange läuft bis der gleiche Bitvektor generiert wird, sind die Anzahlen der Brüder immer endlich. Da $shift$ die Anzahl der Einsen nicht verändert, $newShift$ immer eine zusätzliche 1 generiert und wieder nur solange aufgerufen wird bis der gleiche Bitvektor generiert wird, ist auch die Tiefe des Baumes endlich (endlicher Bitvektor $\Rightarrow$ endlich viele Einsen). Man erhält also einen Baum mit endlicher Tiefe und endlicher Breite, woraus sich die Termination des FRTPS–Algorithmus erschließt. Wieso es sich tatsächlich um einen Baum handelt, der (*) und (**) erfüllt, wird später noch erläutert.

Analysieren wir diesen Algorithmus an einem Beispiel:
Betrachtet man eine drei elementige Grundmenge, also einen dreidimensionalen Bitvektor und startet mit FRTPS(000). In Abbildung 4.1 kann man einen Baum sehen, der von dem FRTPS–Algorithmus durchlaufen wird. Hier hat jedes Kind zu seinem Vater nur ein Bit Unterschied (***)[3]. Dieser Baum wird zuerst in der Tiefe und dann von links nach rechts durchlaufen. Das am weitesten links stehende Kind eines Vaters, wird durch $newShift$ erzeugt. Seine Brüder werden dann aus ihm durch einen $shift$ generiert.
Man kann somit diesen Algorithmus als Durchlauf durch einen Baum verstehen, welcher die Beziehung (***) und somit die Ordnung (*) erfüllt. Was im Folgenden gezeigt wird ist, dass der Algorithmus sowohl die Ordnung (**) erfüllt, als auch alle möglichen Bitvektoren aufzählt und jeder Bitvektor nur einmal vorkommt.

[3](***) ist eine Markierung für diese Eigenschaft

Algorithm 1 Der FRTPS(a)

Require: $a = 0$.
Ensure: Potenzmenge von a.
  $b \leftarrow a$
  {Erzeugt eine neue Ebene in dem Baum}
  $a \leftarrow newShift(a)$
  if $a == b$ **then**
    return $\emptyset$
  end if
  $Result \leftarrow \emptyset$
  {Erzeugt alle Brüder von a}
  repeat
    $Result \leftarrow Result \cup \{a\}$
    {gehe zunächst in die Tiefe des Baumes}
    $Result \leftarrow Result \cup FRTPS(a)$
    $b \leftarrow a$
    {erzeuge einen Bruder (links nach rechts)}
    $a \leftarrow shift(a)$
  until $a \neq b$
  return $Result$

Beweis:
Betrachtet man nun den Algorithmus als eine Abbildung, die im i–ten Schritt eine natürlichen Zahl erzeugt (2–adische Darstellung). Der Definitionsbereich der FRTPS–Abbildung ist eine echte Teilmenge von $\mathbb{N}$ (da Termination bereits Bewiesen) und der Wertebereich entspricht $\{0..2^n - 1\}$.
Stellt man sich wieder den Baum vor, so kann man immer einen Weg von der Wurzel zu einem konkreten Bitvektor b finden. Es ist leicht zu erkennen, dass in der Tiefe Eins, die erste 1 (von rechts) von b erzeugt wird. In der Tiefe Zwei wird die zweite 1 (von rechts) von b erzeugt und so weiter. So wird also nach und nach der gewünschte Bitvektor erzeugt (Surjektivität). Da zusätzlich, wie bereits gezeigt wurde, der entsprechende Baum endliche Tiefe und Breite hat, wird auch tatsächlich irgendwann dieser Bitvektor generiert (muss wegen Tiefendurchlauf betrachtet werden).
Wie schaut es nun mit der Injektivität aus? Dazu betrachtet man wiederum den Baum und wendet die strukturelle Induktion an. Das Ziel ist es, über die Surjektivität und über die gleiche Kardinalität von Bild– und Urbildbereich auf die Injektivität zu schließen (nur möglich bei endlicher Kardinalität).

(Die Induktionsannahme) Der Definitionsbreich hat 2^n Elemente.

(Induktion über) n, die länge der Bitvektoren.

(Der Induktionsbeginn) Starten wir nun mit dem einelementigen Bitvektor, also $n = 1$: So generiert die FRTPS–Abbildung zwei Bitvektoren ($2^1 = 2$).

(Der Induktionsschritt) $n \rightarrow n + 1$:
Betrachtet man den n–elementigen Fall und den dazugehörigen Baum, so kann man diesen leicht zu dem $n + 1$ elementigen Fall erweitern. Zur weiteren Erklärung dienen die Beispiele in Abbildung 4.1 und Abbildung 4.2. Am Anfang des FRTPS–Algorithmus steht ein $newShift$ und später ein rekursiver Aufruf, weshalb der Baum zunächst in der Tiefe durchlaufen wird. Kann durch den $newShift$ kein neuer Bitvektor erzeugt werden, so können seine Brüder durch den $shift$ erzeugt werden. Zu beachten ist, dass bei dem Erzeugen seiner Brüder wieder in die Tiefe gegangen wird (rekursiver Aufruf). Zusammenfassend heißt dies: Es gilt so lange wie möglich durch $newShift$ neue Bitvektoren zu erzeugen, geht dies nicht mehr, so ist mit $shift$ fortzufahren, bis $newShift$ wieder in der Lage ist neue Bitvektoren zu erzeugen, und so weiter.
Startet wir mit dem dreielementigen Fall (Abb. 4.1), welchen wir zu dem vierelementigen Fall (Abb. 4.2) erweitern werden. Bei dem vierelementigen Fall wird zunächst genau der gleiche Baum wie bei dem dreielementigen durchlaufen, wobei hinter jedem Bitvektor eine 0 geschrieben wird (entspricht der roten 0 in Abb. 4.2). Da nun in der ersten Ebene von Abbildung 4.2 das am weitesten rechts stehende Kind nicht mehr 1 (als natürliche Zahl zu lesen) entspricht, wird dieses als nächstes generiert (oberster Bitvektor mit grüner Zahl in Abb. 4.2). Schließlich wird zum letzten Mal Abbildung 4.1 durchlaufen, wobei am Ende eines Bitvektors immer eine 1 angefügt wird.
Dass dies kein Zufall ist, leitet sich von $shift$ und $newShift$ ab, denn diese Operationen verändern immer nur die am weitesten links stehende 1, oder die erste 0.

(Der Induktionsschluss) Da der Baum des $n + 1$–Falles auf zwei Bäume des n–Falles reduziert werden kann, folgt, dass die Abbildung $2 * 2^n = 2^{n+1}$ Elemente aufzählt.

Beweisende der Induktion ■

Da der Wertebereich ebenso wie der Definitionsbereich 2^n Elemente beinhaltet und die Surjektivität gilt, muss jeder Bitvektor genau einmal generiert worden sein. Anders ausgedrückt, die Abbildung ist injektiv.

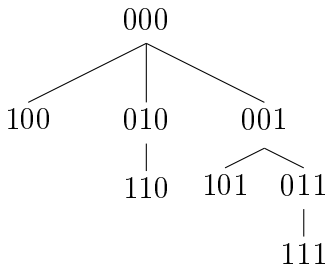

Abbildung 4.1.: Generierter Baum des FRTPS Algorithmus. Er wird in der Tiefe und dann von links nach rechts durchlaufen.

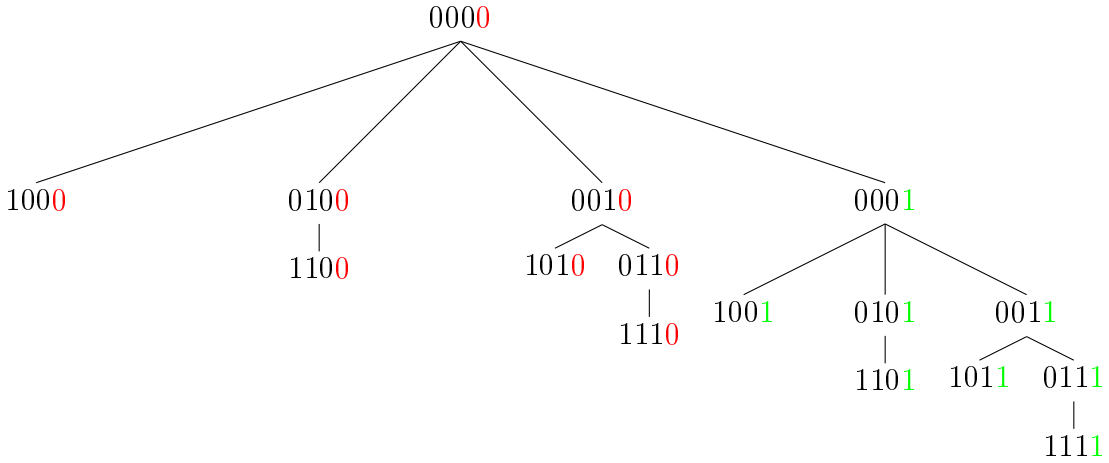

Abbildung 4.2.: $n \to n+1$

Nun bleibt noch zu zeigen, dass der FRTPS–Algorithmus der Ordnungsbedingung (**) genügt. Dazu wähle ich an dieser Stelle eine andere Darstellungsform zur besseren Veranschaulichung (Abb. 4.3). Diese Darstellungsform entspricht der Reihenfolge, in der die Bitvektoren erzeugt wurden (zuerst in die Tiefe und dann von links nach rechts). Wie man an der Abbildung sieht, ist in diesem Fall das Komplement vom i–ten Bitvektor an der Stelle $2^n - 1 - i$ zu finden. Nun stellt sich die Frage ob dies immer der Fall ist? Die Antwort lautet ja. Zunächst einmal gilt, dass der konstante $\mathbb{1}$–Vektor immer zuletzt erzeugt wird. Betrachtet man wieder den dazugehörigen Baum, so stellt man fest, dass es das am weitesten rechts stehende Blatt ist. Da der Baum von oben nach unten und von links nach rechts durchlaufen wird, ist der konstante $\mathbb{1}$–Vektor das Blatt, welches

zuletzt betrachtet wird. Vertauscht man bei dem FRTPS–Algorithmus die Rollen von 0 und 1 (bzw. vertauscht man bei den Definitionen 1 und 0), so macht dies natürlich keinen großen Unterschied. Hier notieren wir mit $shift'$ und $newShift'$ die neuen Operationen, die bei der Vertauschung der Rollen von 1 und 0 entstehen. Hat man einen Bitvektor a und sein Komplement b, so stellt man schnell fest, dass $shift(a)$ das Komplement von $shift'(b)$ ist. Für $newShift$ und $newShift'$ gilt selbiges.
Wir haben also einen FRTPS–Algorithmus der mit dem 0–Bitvektor beginnt und einen FRTPS'–Algorithmus der mit dem $\mathbb{1}$–Bitvektor beginnt. Da der 0–Bitvektor das Komplement des $\mathbb{1}$–Bitvektors ist, und die entsprechenden Operationen die Komplementäreigenschaft nicht zerstören, produzieren diese Algorithmen in jedem Schritt jeweils das Komplement (Abb. 4.3). Dies heißt wiederum, dass der FRTPS–Algorithmus die Ordnungsbedingung (**) erfüllt.
Zusammenfassend wurde gezeigt, dass der FRTPS–Algorithmus die geforderten Ordnungen (*) (s. z.B. Abb. 4.2) sowie (**) (s. z.B. 4.3) erfüllt und dass jeder Bitvektoren genau einmal aufgezählt wurde.

↓ 000
100
010
110
001
101
011
↑ 111

Abbildung 4.3.: Der Reihe nach erzeugte Bitvektoren. Durchgehend von oben und unten, sind die Elemente jeweils das Komplement.

Beweisende des gesamten Beweises ■

Zur Erinnerung: Das Ziel ist es eine optimale Partition, bezüglich eines Clusterkriteriums zu berechnen. Dazu haben wir zunächst die Potenzmenge auf die Menge aller n-stelligen Bitvektoren, mit einer bijektiven Abbildung abgebildet. Auf der Menge aller n–stelligen Bitvektoren wurde eine Ordnung geschaffen, die den Bedingungen (*) und (**) genügt. Damit wurde zudem auch eine Ordnung gefunden, die entsprechend die Potenzmenge ordnet. Durch die Ordnungsbedingung (*) auf der Potenzmenge ist man in der Lage, minimale Aktualisierungen durchzuführen. Durch die Ordnungsbedingung (**) ist man in der Lage, doppelte Berechnungen gleicher Partitionen zu verhindern. Dies geschieht, in dem eine kleine Modifikation des FRTPS–Algorithmus durchgeführt wird und so ein optimaler Algorithmus für unsere Zwecke geschaffen wird. Das einzige was gemacht werden

muss, ist einen Zähler einzubauen, der nach $2^{n-1} - 1$ Schritten eine vorzeitige Termination bewirkt (verhindert die Aufzählung von komplementären Mengen, welche bereits abgearbeitete Partitionen darstellen). Graphisch gesehen fällt der rechte Teilbaum mit der Wurzel 1 (als natürliche Zahl zu interpretieren) weg, somit muss man nicht einmal mitzählen.
Ein anderer Lösungsansatz stammt von Prof. Dr. Ritter, der als Grundidee einen Hyperwürfel benutzt. Dieser wird hier nicht näher erläutert, jedoch kann man die Grundidee auch bei dem von mir entwickelten Algorithmus anwenden. Dazu betrachtet man einen n–dimensionalen Hyperwürfel. Jede Dimension entspricht einem Bit, läuft man in einer Dimension eine Kante entlang, so ändert sich das jeweilige Bit. Die Ecken entsprechen also immer einem Bitvektor, wobei sich zwei verbundene Ecken immer nur um ein Bit unterscheiden. Durch **unterschiedliche** Pfade kann man so zu einem Punkt gelangen. Dies wird bei dem FRTPS–Algorithmus durch die Bewegungseinschränkung in „niederen“ Dimensionen verhindert. Damit ist die Einschränkung gemeint, die es erlaubt, nur an Kanten entlang zu laufen die in eine Dimension führen, deren Index kleiner ist als jeder der bisherigen entlanggelaufenen. Betrachtet man den Knoten 00110, so darf man in den Dimensionen eins und zwei entlanglaufen, da der bisherig kleinste Index einer Dimension drei ist (von links betrachtet befindet sich die erste 1 an der dritten Position).

4.5. Die Implementierungsdetails

Der Rückgabewert des FRTPS–Algorithmus ist für uns wie erwähnt irrelevant. Wichtig ist jedoch, wie dieser durch die Potenzmenge läuft. Jedes Element in dem Baum entspricht einer möglichen Partition. Das Kind eines Knotens entspricht der Partition, bis auf eine Neuzuordnung eines Datenpunktes, was uns nun ermöglicht eine Aktualisierung des Clusterkriteriums zu berechnen. In der ersten Ebene berechnet man die Clusterkriterien auf herkömmliche Art und Weise und speichert diese in jedem Knoten. Bei den restlichen Knoten ist es somit möglich, wie bereits erwähnt, eine Aktualisierung des Vaters durchzuführen, was den Algorithmus beschleunigt.
Um zusätzliche Geschwindigkeit zu erlangen, könnte man die erste Ebene berechnen und den letzten Knoten auslassen, anstatt von Anfang an in die Tiefe des Baumes zu gehen. Der letzte Knoten ist überflüssig, da bereits bei $2^{n-1} - 1$ Knoten abgebrochen wird. In der so entstandenen ersten Ebene, kann man nun für jeden Knoten einen Thread generieren und den unmodifizierten FRTPS–Algorithmus laufen lassen. Diese Threads benötigen keinerlei Kommunikation untereinander, was wiederum eine sehr effiziente, parallele Ausführung ermöglicht. Erst bei der Termination aller Threads ist es nötig, das globale Optimum aus den lokalen Optima der Threads zu berechnen.
Es stellt sich die Frage, weshalb es besser ist, in die Tiefe, anstatt in die Breite des Baumes zu gehen. Dies hat wiederum mit dem Speicherverbrauch zu tun. Geht man

in die Breite, so muss man sich alle Knoten in einer Ebene merken, um zur nächsten Ebene zu gelangen. Wie man weiß, verursacht dies exponentiellen Speicherverbrauch, da die Breite eines Baumes exponentiell steigt. Bei dem Tiefendurchlauf merkt man sich hingegen nur den aktuellen Knoten und die Clusterkriterienwerte der Väter. Somit ist der Speicherbedarf nur linear.

4.6. Die Zusammenfassung

In diesem Kapitel wurde ein Algorithmus dargestellt, mit dessen Hilfe man für n bis etwa 35, mit einem heutigen Rechner, exhaustiv die beste Partition bezüglich eines Clusterkriteriums berechnen kann. Die Stärken des Algorithmus liegen im linearen Speicherverbrauch, in der Einsparung von Rechenzeit durch Aktualisierungen, dem Verhindern von redundanten Berechnungen (Stichwort Komplemente) und der Parallelisierbarkeit. Trotz aller Optimierungen bleibt dieser Algorithmus exponentiell in der Laufzeit und ist daher nur begrenzt einsetzbar. Auch die Clusteranzahl ist auf zwei beschränkt.

Kapitel 5.

Spektrales Clustern

Nun zu dem Kernthema dieser Arbeit. Wie bereits erwähnt ist es NP–hart, eine optimale Partition bezüglich eines Clusterkriteriums zu berechnen. Deshalb wird eine Approximation nötig, die hier mit Hilfe von Eigenwerten und Eigenvektoren getroffen wird. Der Name des spektralen Clusterns kommt gerade von diesen Eigenwerten, denn die Menge aller Eigenwerte wird auch als Spektrum bezeichnet.
Zunächst werden spezielle Matrizen und Vektoren eingeführt, die es erlauben, die Clusterkriterien umzuschreiben. An diese Vektoren sind bestimmte Bedingungen geknüpft (eine Art **Codierung** der Cluster), welche zu einem späteren Zeitpunkt fallen gelassen werden (Relaxation). Dies ist der entscheidende Schritt. So wird auf einmal das NP–harte Problem, zu einem relativ einfachen Problem. Es müssen respektiv die kleinsten oder größten Eigenvektoren berechnet werden. Hat man diese Eigenvektoren von den speziellen Matrizen berechnet, so muss man als letzten Schritt die Partition aus den Eigenvektoren gewinnen (**Decodierung**). Tatsächlich geschieht dies durch ein anderes Clusterverfahren, worauf später noch eingegangen wird.

5.1. Wichtige Definitionen und Sätze

Wie bereits erwähnt benutzt man spezielle Matrizen um die Clusterkriterien umzuschreiben. Genauer gesagt, betrachtet man die Clusterkriterien als eine Summe von Rayleigh–Quotienten, welche später noch eingeführt werden.

5.1.1. Der ungerichtete, positiv gewichtete Graph der Ähnlichkeitsmatrix

Bei dem Betrachten eines Merkmals–Datensatzes der geclustert werden soll, ist der erste Schritt aus den Merkmalen eine Ähnlichkeit $s_{i,j}$ zu gewinnen. Der nächste Schritt wäre,

aus der Ähnlichkeit einen ungerichteten, gewichteten Graphen $G = (V, E, f)$ zu generieren. Eine triviale Methode wäre es, die Ähnlichkeit einfach als einen ungerichteten, positiv gewichteten Graphen aufzufassen. Man hat also einen vollständigen ungerichteten, positiv gewichteten Graphen (maximale Kantenanzahl), wobei anzumerken ist, dass dieser auch selbstreferenzierende Kanten hat (jeder Knoten hat eine Kante zu sich selbst).
Es gibt viele weitere Methoden einen ungerichteten, positiv gewichteten Graphen zu generieren. In der Kombinatorik geht man meist von Graphen aus, die keine selbstreferenzierenden Kanten haben. Man könnte also $E := \{\{i,j\}|i,j \in V, i \neq j$, mit $s_{i,j} > 0\}$ und $f(i,j) := s_{i,j}$ setzen.
In Luxburg [13] wird des weiteren von einem ϵ–Graphen, einem *k–nearest Neighborhood* Graphen und einem *mutual k–nearest Neighborhood* Graphen gesprochen. Bei dem ϵ–Graphen setzt man $E := \{\{i,j\}|i,j \in V$ mit $s_{i,j} > \epsilon\}$ mit $f(i,j) := s_{i,j}$. Oftmals geht man auch bei einem ϵ–Graphen von einem ungewichteten Graphen aus ($f(i,j) = 1$ falls $s_{i,j} > \epsilon$).
Sucht man zu jedem Knoten die k nächsten Nachbarn und löscht alle anderen Kanten die nicht zu diesen führen, so erhält man einen gerichteten, positiv gewichteten Graphen (Abb. 5.2). Das heißt, es gilt nicht unbedingt mehr $f(i,j) = f(j,i)$. Um dies zu beheben kann man entweder gerichtete Kanten löschen ($E_{\text{neu}} := \{\{i,j\}|(i,j),(j,i) \in E_{\text{alt}}\}$), oder Kanten einsetzen ($E_{\text{neu}} := \{\{i,j\}|(i,j)$ oder $(j,i) \in E_{\text{alt}}\}$). Bei dem durch Löschen von Kanten erzeugten ungerichteten, positiv gewichteten Graphen spricht man von dem *k–nearest Neighborhood* Graphen (Abb. 5.4). Bei dem durch Hinzufügen von Kanten entstandenen ungerichteten, positiv gewichteten Graphen von einem *mutual k–nearest Neighborhood* Graphen (Abb. 5.3).
Sowohl beide *k–nearest Neighborhood* Graphen als auch der ϵ–Graph haben etwas gemeinsam: sie besitzen beide einen frei wählbaren Parameter, k beziehungsweise ϵ. Dieser Parameter kann ein sehr effektives Werkzeug sein, aber auch eine große Schwachstelle. Einerseits kann man ihn so einstellen, dass es bei Versuchen zu den erwünschten Ergebnissen führt, andererseits hat er jedoch keine allgemeinere Bedeutung (Überanpassung).

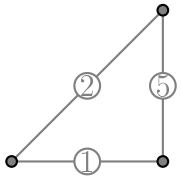

Abbildung 5.1.: Ungerichteter, positiv gewichteter Ausgangsgraph

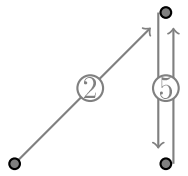

Abbildung 5.2.: 1–gerichteter, positiv gewichteter Nachbarschaftsgraph entstanden aus Abbildung 5.1

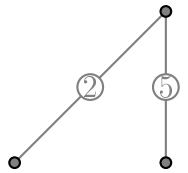

Abbildung 5.3.: mutual 1–nearest Neighboorhood Graph

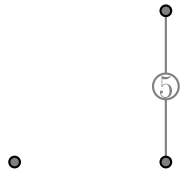

Abbildung 5.4.: 1–nearest Neighboorhood Graph

5.1.2. Die Gewichtsmatrix des Graphen

Wir haben nun einen ungerichteten, positiv gewichteten Graphen den wir clustern wollen. Die betreffenden Informationen zu diesem Graphen werden in einer Matrix W gespeichert.

Definition 5.1 *Sei $G = (V, E, f)$ ein ungerichteter, positiv gewichteter Graph mit Gewichtsfunktion $f : E \to \mathbb{R}_+$. Dann ist*

$$W(i,j) := w_{i,j} = \begin{cases} f(i,j), & \textit{falls } \{i,j\} \in E \\ 0, & \textit{sonst} \end{cases}.$$

5.1.3. Die Degreematrix

Wenn man an den *NormalCut* zurück denkt, so steht im Nenner das Volumen eines Clusters. Dieses Volumen ist die Summe der Grade aller Knoten innerhalb eines Clusters.

Deshalb müssen diese Informationen auch in einer Matrix D gespeichert werden. Zur Erinnerung: d_i entspricht dem Grad (Summe aller Kanten) des Knotens i.

Definition 5.2 *(s. Luxburg [13]) Die Gradmatrix D ist eine Diagonalmatrix mit*

$$D(i,j) := \begin{cases} d_i, & \textit{falls } i = j \\ 0, & \textit{sonst} \end{cases}.$$

Auf der i–ten Position in der Diagonalen steht also der Grad des i–ten Knotens.

5.1.4. Die Laplacematrix

In der Literatur gibt es mehrere ähnliche Definitionen der Laplace–Matrix. Es werden drei $(L, L_{\text{sym}}, L_{\text{rw}})$ verschiedene Laplacematrizen definiert. Ist hier die Rede von der Laplacematrix, so ist stets L gemeint.

Definition 5.3 *(s. Luxburg [13]) Die Laplacematrix, oder auch unnormalisierte Laplacematrix entspricht:*

$$L := D - W.$$

Anzumerken ist, dass die Laplacematrix unabhängig von der Diagonalen der Gewichtsmatrix und somit unabhängig von selbstreferenzierenden Kanten ist. Des weiteren wird folgender Satz benötigt, um später die Laplacematrix in Beziehung mit dem *RatioCut* zu setzen:

Satz 5.1 *(s. Luxburg [13]) Die Laplacematrix L erfüllt folgende Eigenschaften:*

1. *Für jeden Vektor $f \in \mathbb{R}^n$ gilt folgende Gleichung*
$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$
2. *L ist symmetrisch und positiv semi–definit.*
3. *Der kleinste Eigenwert von L ist 0, und der dazugehörige Eigenvektor ist der konstante Einsvektor $\mathbb{1}$.*
4. *L hat n nicht–negative, reellwertige Eigenwerte $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.*

Beweisführung:
Part(1): Aus der Definition von d_i folgt:

$$f^T L f = f^T D f - f^T W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$
$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2\right) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Part(2): Die Symmetrie von L folgt direkt aus der Symmetrie von W und D. Die positiv Semi–definitheit folgt direkt aus (1).
Part(3): Folgt offensichtlich aus (1).
Part(4): Folgt aus der Symmetrie der Matrix, der Reellwertigkeit und der Anzahl der Eigenwerte. Die positiv Semi–definitheit ermöglicht die untere Abschätzung.

■

5.1.5. Die normalisierte Laplacematrix

Unter dem Begriff der normalisierten Laplacematrizen versteht man zweierlei Definitionen, die in Beziehung zueinander stehen.

5.1.5.1. Die normalisierte, symmetrische Laplacematrix

Definition 5.4 *(s. Luxburg [13]) Die normalisierte, symmetrische Laplacematrix ist*

$$L_{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}.$$

Die Laplacematrix, wie auch die normalisierte, symmetrische Laplacematrix sind symmetrisch und haben dadurch einige für uns sehr nützliche Eigenschaften. Da die Matrizen aus dem $\mathbb{R}^{n\times n}$ stammen, weiß man aus der Linearen Algebra 2 (s. Ritter [21]):

- Das charakteristische Polynom χ hat Dimension n.
- Alle Eigenwerte sind reell und somit auch alle Eigenvektoren.
- Alle Eigenvektoren stehen bezüglich des Standardskalarprodukts aufeinander senkrecht.
- Ein Unterraum, der aus den Eigenvektoren eines Eigenwertes gebildet wird, hat die Dimension der Vielfachheit des Eigenwertes.

Daraus folgt, dass der Raum in die direkte Summe der Eigenräume zerfällt und man somit eine ON–Basis gefunden hat, die aus den Eigenvektoren besteht. Schreibt man nun alle Eigenvektoren spaltenweise in eine Matrix Y und die dazugehörigen Eigenwerte in der gleichen Reihenfolge in eine Diagonalmatrix D, so gilt für eine symmetrische Matrix S der **Hauptsatz für symmetrische Matrizen**:

$$SY = YD \text{ oder äquivalent die } \textbf{Spektraldarstellung } S = YDY^{-1} = YDY^{T}.$$

Durch die Spektraldarstellung gelingt es uns, ein n–dimensionales Problem, zum Beispiel ein Gleichungssystem, in n eindimensionale Probleme zu zerlegen. Wie bereits geschildert besteht Y aus einer Basis, wodurch die Spalten linear unabhängig sind (Injektivität). Die Spalten spannen den $\mathbb{R}^n$ auf (Surjektivität), wodurch Y ein Automorphismus ist. Genauer gesagt entspricht Y sogar einer Drehung oder einer Drehspiegelung.

5.1.5.2. Die Irrfahrt

Der *NormalCut* hat eine enge Beziehung zu L_{sym}, welche wiederum eine enge Beziehung zu L_{rw} hat. $I - L_{\text{rw}}$ ist dabei die Gewichtsmatrix eines speziellen Graphens. Diesen gerichteten, positiv gewichteten Graphen kann man als Irrfahrt interpretieren, womit man eine Verbindung zu den Markoff–Ketten gefunden hat.

Definition 5.5 *(s. Luxburg [13]) Die normalisierte Laplacematrix einer Irrfahrt entspricht*

$$L_{\text{rw}} := D^{-1}L = I - D^{-1}W.$$

Um eine Verbindung zu einer Irrfahrt herzustellen, konvertiert man den zugrunde liegenden ungerichteten, positiv gewichteten Graphen (den Graphen den wir clustern wollen) in eine Irrfahrt (s. Luxburg [13]). Der resultierende Graph ist nun nicht mehr ungerichtet, wodurch die Übergangsmatrix $P := p_{i,j} := w_{i,j}/d_i$ nicht mehr symmetrisch ist. Formal ausgedrückt gilt

$$P = D^{-1}W.$$

Zunächst ist anzumerken, dass immer nur endlich viele Datenpunkte vorhanden sind und somit ein endlicher Graph besteht (endlich viele Knoten und Kanten). Ist der zu clusternde ungerichtete, positiv gewichteter Graph zusammenhängend, wovon immer

ausgegangen wird, so ist auch der durch P induzierte Graph (die Irrfahrt) zusammenhängend. Man kommt von jedem Knoten in endlicher Zeit mit positiver Wahrscheinlichkeit zu jedem anderen Knoten, da man jede einfache Kante durch zwei Kanten ($\rightleftarrows$) ersetzt. Somit hat man einen endlichen, rekurrenten Graphen vor sich. Das garantiert, dass es ein Gleichgewicht (**GGW** s. Ritter [20]) $\pi = (\pi_1, ..., \pi_n)$ mit $\pi_i = d_i/vol(G)$ gibt (s. Luxburg [13]). Anschaulich dargestellt ist $1/\pi_i$ die erwartete, erste Wiederkehrzeit.
Die Tatsache, dass es ein *GGW* gibt und wir wissen wie jenes aussieht, ermöglicht es uns später eine Veranschaulichung des *NormalCut* darzulegen.
Man kann L_{rw} mit P leicht in Beziehung setzen: $L_{\text{rw}} = I - P$. Als Konsequenz ist v Eigenvektor mit Eigenwert λ von L_{rw}, genau dann wenn $1 - \lambda$ Eigenwert mit Eigenvektor v von P ist (s. Luxburg [13]).

5.1.5.3. Der Zusammenhang von L_{sym} und L_{rw}

Im nachstehenden Satz wird der Zusammenhang zwischen L_{sym} und L_{rw} verdeutlicht. Außerdem wird dieser Satz benötigt, um eine Verbindung zwischen den normalisierten Laplacematrizen (L_{sym}, L_{rw}) und dem *NormalCut* aufzuzeigen.

Satz 5.2 *(s. Luxburg [13]) Die symmetrische Laplacematrix L_{sym} erfüllt folgende Eigenschaften:*

1. *Für jeden Vektor $f \in \mathbb{R}^n$ gilt die Gleichung*
$$f^T L_{\text{sym}} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 .$$
2. *v ist ein Eigenvektor mit Eigenwert λ von L_{rw} genau dann, wenn λ ein Eigenwert von L_{sym} mit Eigenvektor $w = D^{1/2}v$.*
3. *v ist ein Eigenvektor mit Eigenwert λ von L_{rw} genau dann, wenn v das verallgemeinerte Eigenproblem $Lv = \lambda Dv$ löst.*
4. *0 ist ein Eigenwert von L_{rw}, bezüglich dem konstanten Einsvektor $\mathbb{1}$ als Eigenvektor. 0 ist ein Eigenwert von L_{sym}, bezüglich dem Eigenvektor $D^{1/2}\mathbb{1}$.*
5. *L_{rw} und L_{sym} sind beide positiv semi–definit und haben n nicht–negative, reellwertige Eigenwerte $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.*

Beweisführung:
Part(1): Geht ähnlich wie Beweis (1) von Satz 5.1.
Part(2): Folgt sofort, wenn man bei der Eigenwertgleichung $L_{\text{sym}}w = \lambda w$ mit $D^{-1/2}$ von links multipliziert und $v = D^{-1/2}w$ substituiert.
Part(3): Folgt sofort, wenn man bei der Eigenwertgleichung $L_{\text{rw}}v = \lambda v$ mit D von links

multipliziert.
Part(4): Folgt sofort aus (1) und (2).
Part(5): Die Behauptung für L_{sym} folgt sofort aus (1), und die Behauptung für L_{rw} folgt sofort aus (2).

■

5.1.6. Der Rayleigh–Quotient

Nun zu einer zentralen Definition. Mit Hilfe des Rayleigh–Quotienten werden die Clusterkriterien umgeschrieben und später wird eine Verbindung zu Eigenvektoren hergestellt.

Definition 5.6 *(s. Chung [4]) Sei* $x \in \mathbb{R}^n \backslash \{0\}$ *und* $A \in \mathbb{R}^{n \times n}$*, dann ist der Rayleigh–Quotient*

$$R_A(x) = \frac{x^T A x}{x^T x}.$$

5.1.6.1. Die Diagonalmatrizen, Eigenvektoren und Eigenwerte

Ist nun die Matrix $A := a_{i,j}$ von $R_A(x)$ in Diagonalgestalt und wollten wir $R_A(x)$ minimieren, so kommt man zu folgendem Ergebnis:

$$\begin{aligned} x \text{ ist Minimalstelle von } R_A(x) = \frac{x^T A x}{x^T x} &= \frac{1}{||x||^2} < x, Ax > = < \frac{x}{||x||}, \frac{Ax}{||x||} > \\ \Leftrightarrow x \text{ ist Minimalstelle von } \textstyle\sum_{i=1}^n x_i^2 a_{i,i} &\text{ wobei } \textstyle\sum_{i=1}^n x_i^2 = 1 \\ \Leftrightarrow x = e_l &\text{ wobei } a_{l,l} \text{ minimal.} \end{aligned}$$

Hier ist aber gerade e_l ein Eigenvektor bezüglich des Eigenwertes $a_{l,l}$ von A. Sei nun $A(i,i) = a_i = a_{i,i}$ aufsteigend geordnet. Möchte man das $min_x R_A(x)$ berechnen, wobei $x \perp e_1$ steht, so wird das Minimum bei e_2 erreicht. Möchte man, dass $x \perp e_1$ und $x \perp e_2$ steht, so wird das Minimum bei e_3 erreicht, und so weiter.
Man weiß, an welcher Stelle der jeweilige Minimalwert angenommen wird und kann auch den Wert

$$\frac{< x, Ax >}{< x, x >} = \frac{< x, a_i x >}{< x, x >} = a_i \frac{< x, x >}{< x, x >} = a_i$$

bestimmen, da x Eigenvektor zum Eigenwert a_i ist.
Zusammengefasst gilt, wenn die Matrix A in Diagonalgestalt ist, wird das Minimum von $R_A(x)$ in einem Eigenvektor erreicht und hat den Wert des dazugehörigen Eigenwertes.

5.1.6.2. Die symmetrischen Matrizen, Eigenvektoren und Eigenwerte

Wenn A eine reguläre symmetrische Matrix ist, dann gilt $A = Y A_{\text{diag}} Y^T$, wobei die Spalten von Y Eigenvektoren von A sind und A_{diag} die Diagonalgestalt von A ist. Des weiteren wissen wir bereits, dass Y ein Automorphismus ist. Will man nun $R_A(x)$ minimieren, so kann man ebensogut $R_{A_{\text{diag}}}(x)$ minimieren und die Ergebnisse umrechnen. Man substituiert $y = Yx$ und kommt somit auf

$$\begin{aligned} R_A(y) = R_A(Yx) = \frac{(Yx)^T Y A_{\text{diag}} Y^T Y x}{(Yx)^T Y x} = \frac{x^T Y^T Y A_{\text{diag}} Y^T Y x}{x^T Y^T Y x} \\ = \frac{x^T A_{\text{diag}} x}{x^T x} = R_{A_{\text{diag}}}(x). \end{aligned}$$

Nachdem der symmetrische Fall auf den Diagonalfall zurückgeführt wurde, stellt man fest, da Y aus den Eigenvektoren von A besteht und die Lösung von $R_{A_{\text{diag}}}(x)$ Einheitsvektoren (e_i) sind, dass die Lösung $y = Ye_i$ von $R_A(y)$ **wieder aus Eigenvektoren** besteht.
Seien die Eigenvektoren v_i der Größe ihrer Eigenwerte nach sortiert, so kann festgestellt werden, dass das Minimum von $R_A(x)$ bei dem ersten Eigenvektor v_1 zu finden ist und den Wert des dazugehörigen Eigenwertes annimmt. Will man nun, dass $y \perp v_1$ steht, so findet man das Ergebnis in v_2 (**genau wie in dem Diagonalfall**), und so weiter.

5.1.7. Der verallgemeinerte Rayleigh–Quotient

Es gibt eine verallgemeinerte Version des Rayleigh–Quotienten, der später für den *MinMaxCut* und *NormalCut* benötigt wird.

Definition 5.7 *[1] Seien nun $x \in \mathbb{R}^n \backslash \{0\}$, $A, B \in \mathbb{R}^{n \times n}$ symmetrische Matrizen und B regulär, dann ist der Rayleigh–Quotient*

$$R_{A,B}(x) = \frac{x^T A x}{x^T B x}.$$

Wenn B in Diagonalform ist, so kann man durch Substitution $y = B^{-1/2}x$ auf

$$R_{A,B}(y) = R_{A,B}(B^{-1/2}x) = \frac{(B^{-1/2}x)^T A B^{-1/2}x}{(B^{-1/2}x)^T B B^{-1/2}x} = \frac{x^T B^{-1/2} A B^{-1/2}x}{x^T B^{-1/2} B B^{-1/2}x} = \frac{x^T B^{-1/2} A B^{-1/2}x}{x^T x} = R_{B^{-1/2}AB^{-1/2}}(x)$$

zurückführen.

5.2. Der RatioCut und die Laplacematrix

Mit Hilfe der eingeführten Definitionen, kann man jetzt die Clusterkriterien umschreiben. Beginnend mit dem *RatioCut*, wird im Folgenden der *NormalCut* und der *MinMaxCut* betrachtet.
Bei *RatioCut* und *NormalCut* wird zunächst der allgemeine Fall und anschließend der 2–Clusterfall betrachtet.

5.2.1. Der allgemeine Fall

Zur Erinnerung: Hat man einen ungerichteten, positiv gewichteten Graphen $G = (V, E, g)$, mit Adjazenzmatrix w_{ij} und verfolgt das Ziel,

$$\text{argmin}_{A_1,...,A_k}\text{RatioCut}(A_1, ..., A_k), \text{ wobei } (A_i)_{i\in\{1...k\}} \text{ eine Partition ist, zu berechnen.} \tag{1}$$

Betrachtet man die Laplacematrix aus Definition 5.3, den *RatioCut* und die k Dirac Vektoren δ_i ($\delta_{ij} = 1$ wenn Punkt j im i–ten Cluster, 0 sonst), so stellt man fest, dass

$$\delta_i^T L \delta_i = \text{cut}(A_i, \overline{A}_i)$$

und somit

$$\text{RatioCut}(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{\delta_i^T L \delta_i}{\delta_i^T \delta_i} \text{ ist.}$$

Der *RatioCut* wurde aus einer Summe von Rayleigh–Quotienten geschrieben, wobei die Laplacematrix L verwendet wurde.

5.2.2. Der Spezialfall für zwei Cluster

Im allgemeinen Fall wurden die Dirac Vektoren verwendet, um das Problem umzuschreiben. An dieser Stelle wird ein anderer Vektor definiert (s. Luxburg [13]). Der Grund für die Betrachtung des Spezialfalles für zwei Cluster liegt an der Unähnlichkeit des kleinsten Eigenvektors (gemessen an den Eigenwerten), bezüglich L, zu einem Dirac Vektor. Laut Satz 5.1 entspricht dieser Eigenvektor dem konstanten $\mathbb{1}$ Vektor. Die Dirac Vektoren haben jedoch an verschiedenen Stellen eine 0 stehen. Bei dem spektralen Clustern hofft man, dass die Eigenvektoren ähnlich zu den Dirac Vektoren sind (später dazu mehr), wobei sich diese Ähnlichkeit auf die Codierung eines Clusters bezieht. Der $\mathbb{1}$ Vektor enthält jedoch überhaupt keine Informationen über ein Cluster. Deshalb definiert man im 2–Clusterfall die Vektoren anders zu den Dirac Vektoren, um so ein Überspringen des $\mathbb{1}$ Vektors zu ermöglichen.
Sei V die Menge aller Knoten eines ungerichteten, positiv gewichteten Graphens, welcher geclustert werden soll, und $A \subset V$ gegeben, dann definieren wir den Vektor $f \in \mathbb{R}^n$ mit den Einträgen

$$f_i = \begin{cases} \sqrt{|\overline{A}|/|A|}, & \text{falls } v_i \in A \\ -\sqrt{|A|/|\overline{A}|}, & \text{falls } v_i \in \overline{A} \end{cases}. \tag{2}$$

Nun können folgende Umformungen vorgenommen werden und der *RatioCut* mit der Laplacematrix L in Verbindung gesetzt werden:

$$\begin{aligned} f^T L f = & \frac{1}{2} \sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2 \\ = & \frac{1}{2} \left(\sum_{i \in A, j \in \overline{A}} w_{i,j} \left(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}} \right)^2 + \sum_{i \in \overline{A}, j \in A} w_{i,j} \left(-\sqrt{\frac{|\overline{A}|}{|A|}} - \sqrt{\frac{|A|}{|\overline{A}|}} \right)^2 \right) \\ = & \operatorname{cut}(A, \overline{A}) \left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2 \right) \\ = & \operatorname{cut}(A, \overline{A}) \left(\frac{|A| + |\overline{A}|}{|A|} + \frac{|A| + |\overline{A}|}{|\overline{A}|} \right) \\ = & |V| \operatorname{RatioCut}(A, \overline{A}). \end{aligned}$$

Zusätzlich haben wir

$$\sum_{i=1}^{n} f_i = \sum_{i\in A} \sqrt{\frac{|\overline{A}|}{|A|}} - \sum_{i\in \overline{A}} \sqrt{\frac{|A|}{|\overline{A}|}} = |A|\sqrt{\frac{|\overline{A}|}{|A|}} - |\overline{A}|\sqrt{\frac{|A|}{|\overline{A}|}} = 0.$$

Anders ausgedrückt ist der in (2) definierte Vektor orthogonal zu dem konstanten $\mathbb{1}$ Vektor. Genau hier liegt auch der Unterschied zu dem allgemeinen Fall, wo es diese Bedingung nicht gibt.
Weiterhin ist festzustellen, dass

$$||f||^2 = \sum_{i=1}^{n} f_i^2 = |A|\frac{|\overline{A}|}{|A|} + |\overline{A}|\frac{|A|}{|\overline{A}|} = |\overline{A}| + |A| = n$$

ist.
Wir sind also in der Lage das Minimierungsproblem (1) umzuschreiben. So ist es äquivalent zu

$$\text{argmin}_{A\subset V} f^T L f \text{ wobei } f \perp \mathbb{1},\ f_i \text{ ist wie in (2) definiert und } ||f|| = \sqrt{n}. \tag{3}$$

Lässt man das Aussehen des Vektors f_i bei (3) fallen (Relaxation), so entspricht die Lösung dem zweitkleinsten Eigenvektor. Später zu diesem spektralen Ansatz mehr.

5.3. Der NormalCut und die normalisierten Laplacematrizen

Wir haben den *RatioCut* für den allgemeinen Fall und den 2–Clusterfall umgeschrieben. Lässt man das Aussehen der definierten Vektoren beiseite, so reduziert sich das Minimierungsproblem auf die kleinsten k Eigenvektoren bei k Cluster (später dazu mehr). Nun geht man wie bei dem *RatioCut* vor und betrachtet den allgemeinen Fall und anschließend den Spezialfall für den *NormalCut*.

5.3.1. Der allgemeine Fall

Hier ist der Unterschied von *NormalCut* und *RatioCut* zu beachten, der nur in den unterschiedlichen Nennern besteht. Definiert man wie beim *RatioCut* die k Dirac Vektoren, so kann man feststellen, dass

$$\mathrm{vol}(A_i) = \delta_i^T D \delta_i$$

gilt. Man kommt also auf:

$$\mathrm{NormalCut}(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{\delta_i^T L \delta_i}{\delta_i^T D \delta_i}.$$

Erinnert man sich an den Abschnitt über allgemeine Rayleigh–Quotienten, so kommt man auch hier durch eine Substitution $y_i = D^{1/2}\delta_i$ zu dem gewünschten Ergebnis

$$\mathrm{NormalCut}(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{y_i^T D^{-1/2} L D^{-1/2} y_i}{y_i^T y_i} = \sum_{i=1}^{k} \frac{y_i^T L_{\mathrm{sym}} y_i}{y_i^T y_i}.$$

Man kann also den *NormalCut* als Summe von Rayleigh Quotienten ausdrücken, wobei im Zähler die normalisierte, symmetrische Laplacematrix L_{sym} steht.

5.3.2. Der Spezialfall für zwei Cluster

Äquivalent zu dem *RatioCut* wird gezeigt, dass auch hier der kleinste Eigenvektor übersprungen werden kann (s. Luxburg [13]). Es wird wieder ein Vektor $f \in \mathbb{R}^n$ mit

$$f_i = \begin{cases} \sqrt{\dfrac{vol(\overline{A})}{vol(A)}}, & \text{falls } i \in A \\ -\sqrt{\dfrac{vol(A)}{vol(\overline{A})}}, & \text{falls } i \in \overline{A} \end{cases} \text{ definiert.} \tag{4}$$

Durch eine ähnliche Herleitung wie bei dem *RatioCut*, kommt man zu $(Df)^T \mathbb{1} = 0$, $f^T D f = vol(V)$ und schließlich auf $f^T L f = vol(V) * NormalCut(A, \overline{A})$. Will man bezüglich des *NormalCut* clustern, so muss man über den *NormalCut* minimieren, was zu

$$\text{argmin}_A f^T L f \text{ wobei } f \text{ wie in (4) definiert, } Df \perp \mathbb{1} \text{ und } f^T L f = \text{vol}(V) \tag{5}$$

umgeschrieben wird. Das kann durch eine Substitution von $g := D^{1/2} f$ weitergeführt werden und kommt zu dem Ergebnis:

$$\text{argmin}_A \; g^T D^{-1/2} L D^{-1/2} g \text{ wobei } D^{-1/2} g = f \text{ wie in (4) definiert, } g \perp D^{1/2}\mathbb{1} \text{ und } ||g||^2 = \text{vol}(V).$$

Da $D^{-1/2} L D^{-1/2} = L_{\text{sym}}$ ist, kann man die Formel ein weiteres mal umschreiben, und hat somit eine Verbindung zu L_{sym} gefunden.

$$\text{argmin}_A \; g^T L_{\text{sym}} g \text{ wobei } D^{-1/2} g = f \text{ wie in (4) definiert, } g \perp D^{1/2}\mathbb{1} \text{ und } ||g||^2 = \text{vol}(V). \tag{6}$$

Auch hier ist der Unterschied zu dem allgemeinen Fall, dass $g \perp D^{1/2}\mathbb{1}$, welcher der kleinste Eigenvektor (gemessen an den Eigenwerten) bezüglich L_{sym} ist (Satz 5.2).

5.4. Der MinMaxCut und die Adjazenzmatrix

Hier werden wir auf die Betrachtung des Spezialfalles verzichten und uns nur dem allgemeinen Fall widmen (s. Gu et al. [9]). Zur Erinnerung: W entspricht der Gewichtsmatrix und D der Gradmatrix. Es gilt:

$$\text{cut}(A_i, A_i) = \text{vol}(A_i) - \text{cut}(A_i, \overline{A}_i) = \delta_i^T D \delta_i - \delta_i^T (D - W) \delta_i = \delta_i^T W \delta_i$$

wodurch man

$$\text{MinMaxCut}(A_1, ..., A_k) = \sum_{i=1}^k \frac{\delta_i^T L \delta_i}{\delta_i^T W \delta_i} = \sum_{i=1}^k \frac{\delta_i^T (D - W) \delta_i}{\delta_i^T W \delta_i} = \sum_{i=1}^k \frac{\delta_i^T D \delta_i}{\delta_i^T W \delta_i} - k,$$

umschreiben kann. Man definiert $y_i = \frac{D^{-1/2}\delta_i}{||D^{-1/2}\delta_i||_2}$, $Y = [y_1, ..., y_k]$ wodurch $Y^T Y = I$ gilt. Jetzt ist man in der Lage

$$\text{MinMaxCut}(A_1, ..., A_k) = \sum_{i=1}^k \frac{y_i^T y_i}{y_i^T \widehat{W} y_i} - k = \sum_{i=1}^k \frac{1}{\frac{y_i^T \widehat{W} y_i}{y_i^T y_i}} - k$$

(7)

zu schreiben, wobei $\widehat{W} = D^{-1/2}WD^{-1/2}$ ist. Die Rayleigh–Quotienten stehen im Nenner, weshalb sie maximiert werden müssen, um den $MinMaxCut$ minimieren zu können.

5.5. Der Zusammenhang mit Eigenwerten und Eigenvektoren

Wir haben die Clusterkriterien in einer Summe von Rayleigh–Quotienten dargestellt. Die Kunst der Umschreibungen liegt immer in der Definition der Vektoren. Diese Vektoren sind nichts anderes als eine Codierung von möglichen Partitionen. Die Bedingungen (das Aussehen) an diese Vektoren stellt eine Diskretisierung dar, weshalb das Minimierungsproblem NP–hart bleibt. Lässt man jedoch beliebige Vektoren im R^n zu, so reduziert sich das Problem auf die Berechnung von Eigenvektoren.

5.5.1. Relaxation der Bedingungen

Die spektrale Idee ist es, die Bedingungen der definierten Dirac Vektoren fallen zu lassen (Relaxation), und somit das Minimierungsproblem der Clusterkriterien auf das Minimierungs– oder Maximierungsproblem des speziellen, oder des verallgemeinerten Rayleigh–Quotienten zu reduzieren. Da die einzelnen Vektoren immer noch aufeinander senkrecht stehen sollen, entspricht die Lösung bei k Clustern den kleinsten oder größten k Eigenvektoren (bezüglich ihrer Eigenwerte).
Im Folgenden schreibt man ein letztes Mal alle Probleme um.

5.5.1.1. Der RatioCut

Der Fall für zwei Cluster wird zu

$$x \text{ ist Lösung von } \operatorname{argmin}_{f\in\mathbb{R}^n} f^T L f \text{ wobei } f \perp \mathbb{1} \text{ und } ||f|| = \sqrt{n}$$

$$\Leftrightarrow x \text{ ist Lösung von } \operatorname{argmin}_{f\in\mathbb{R}^n} < f, Lf > \text{ wobei } f \perp \mathbb{1} \text{ und } ||f|| = \sqrt{n}$$

$$\Leftrightarrow x \text{ ist Lösung von } \operatorname{argmin}_{f\in\mathbb{R}^n,\ f\perp\mathbb{1}} < \frac{\sqrt{n}}{\sqrt{< f, f >}} f, L \frac{\sqrt{n}}{\sqrt{< f, f >}} f >$$

$$\Leftrightarrow x \text{ ist Lösung von } \operatorname{argmin}_{f\in\mathbb{R}^n,\ f\perp\mathbb{1}} \frac{n}{< f, f >} < f, Lf >$$

$$\Leftrightarrow x \text{ ist Lösung von } \operatorname{argmin}_{f\in\mathbb{R}^n,\ f\perp\mathbb{1}} n\ \frac{<f,Lf>}{<f,f>}$$
$$\Leftrightarrow x \text{ ist Lösung von } \operatorname{argmin}_{f\in\mathbb{R}^n,\ f\perp\mathbb{1}} \frac{<f,Lf>}{<f,f>}. \tag{8}$$

Laut Satz 5.1 ist $\mathbb{1}$ ein Eigenvektor, zum Eigenwert 0 von L. Somit handelt es sich bei der Lösung (8) um den zweitkleinsten Eigenvektor (bezüglich der Eigenwerte), den so genannten Fiedler Vektor (s. Luxburg [13]).
Betrachtet man den allgemeinen Fall und erlaubt statt der Dirac Vektoren ((1) umgeschrieben), beliebige Vektoren aus $\mathbb{R}^n$, wobei diese weiterhin aufeinander senkrecht stehen sollen. So wird

$$\operatorname{argmin}_{A_1,\dots,A_k}\operatorname{RatioCut}(A_1,\dots,A_k) = \sum_{i=1}^{k} \frac{\delta_i^T L \delta_i}{\delta_i^T \delta_i} \qquad (A_i \text{ ist codiert in } \delta_i)$$

$$\text{zu } \operatorname{argmin}_{f_1,\dots,f_k\in\mathbb{R}^n} \sum_{i=1}^{k} \frac{f_i^T L f_i}{f_i^T f_i}, \text{ wobei } f_i \text{ paarweise aufeinander senkrecht stehen.}$$

So reduziert sich das Problem auf die k kleinsten Eigenvektoren (bezüglich der Größe ihrer Eigenwerte), da es sich hierbei um eine Summe von Rayleigh–Quotienten handelt und das Minimum eines Rayleigh–Quotienten bei den kleinsten Eigenvektoren angenommen wird.

5.5.1.2. Der NormalCut

Der *NormalCut* geht analog zum *RatioCut*, weshalb hier auf eine detaillierte Ausarbeitung verzichtet wird.

5.5.1.3. Der MinMaxCut

Der *MinMaxCut* ist etwas komplizierter. Will man ihn minimieren, so muss man den Rayleigh–Quotienten bezüglich $\widehat{W} = D^{-1/2} W D^{-1/2}$ maximieren (s. (7)). Dabei ist anzumerken, dass $\widehat{W}$ nicht unbedingt positiv semi–definit ist. So ist es also unter Umständen möglich, dass $y_i^T \widehat{W} y < 0$ aber nahe 0 ist (s. (7)). Lässt man die Bedingung (die Gestalt) der y_i fallen, so könnten die Summanden des *MinMaxCut* gegen $-\infty$ gehen. Bei der Relaxation des *MinMaxCut* Problems, muss man nicht nur die Gestalt der y_i Vektoren fallen lassen, sondern auch fordern, dass $y_i^T \widehat{W} y > 0$ ist. Nur dann gilt

$$(x_i)_{i\in\{1\dots k\}} \text{ ist Lösung von } \operatorname{argmin}_{y_1,\dots y_k\in\mathbb{R}^n} \sum_{i=1}^{k} \frac{1}{\frac{y_i^T \widehat{W} y_i}{y_i^T y_i}} - k,$$

wobei y_i paarweise aufeinander senkrecht stehen,

$\Leftrightarrow (x_i)_{i\in\{1\dots k\}}$ sind die größten Eigenvektoren bezüglich ihrer Eigenwerte.

5.6. Das Gewinnen der Partition aus den Eigenvektoren

Durch die Relaxation der Bedingungen (das Ignorieren der Gestalt der Vektoren), hat man einen Zusammenhang zu den Eigenvektoren geschaffen. Man hofft, wie erwähnt, dass die Eigenvektoren die Struktur der optimalen Partition tragen. Es stellt sich die Frage wie man aus den Eigenvektoren die Partition generiert.

5.6.1. Der 2–Clusterfall

Kommt man wieder zu dem Spezialfall für zwei Cluster zurück, bei dem nur der zweitkleinste Eigenvektor (bezüglich der Eigenwerte) betrachtet werden muss, der in der Theorie (die geforderte Gestalt) sowohl negative als auch positive Werte beinhaltet. Jede Dimension entspricht, nach der Definition, einem Knoten. Man ordnet also einem Cluster alle Punkte mit positivem Wert zu und dem anderen Cluster die übrigen.

5.6.2. Der Simplex

Das Dekodieren der Eigenvektoren im 2–Clusterfall ist, wie eben gezeigt, nicht weiter komplex. Allgemein ist dies jedoch schwieriger. Stellen wir uns vor, dass wir von einem ungerichteten, positiv gewichteten Graphen, den wir bezüglich eines Clusterkriteriums clustern wollen, die entsprechenden kleinsten oder größten k Eigenvektoren berechnet haben. Hier entspricht jede Dimension in einem Eigenvektor genau einem Knoten. Denk man an die δ_i, so sollte jeder Knoten genau einmal in einem δ_i enthalten sein. Anders ausgedrückt: schreibt man die δ_i spaltenweise in eine Matrix, so ist in jeder Zeile genau ein Eintrag gleich 1, die anderen gleich 0. Da diese Gestalt aber fallen gelassen wurde, und stattdessen Eigenvektoren berechnet wurden, wurde diese Codierung gestört. Zumindest hofft man, dass die Einträge nicht all zu weit verändert wurden. Anschaulich dargestellt, liegen die Knoten an den Ecken eines Simplex.

Ein n–*Simplex* ist das einfachste Polytop, wobei jeder Punkt es in n–Dimensionen erweitert. So ist zum Beispiel ein 1–Simplex ein Geradensegment mit zwei Enden, ein 2–Simplex ein Dreieck mit drei Punkten (Ecken), ein 3–Simplex ein Tetraeder mit vier Punkten (Ecken), und so weiter.
In der Theorie (die Gestalt der Vektoren) für k Cluster, wird jeder Punkt einer Ecke eines $k-1$–Simplex zugeordnet. Durch die Relaxation der Bedingungen, also durch das Zurückführen auf Eigenvektoren, verschwimmen diese Zuordnungen. Die Konsequenz hieraus ist, dass man ein anderes Clusterverfahren (z.B. k–means oder MST (s. Ritter [21])) verwenden muss, um alle Punkte den Ecken wieder zuordnen zu können.
Was bringt es also, ein Clusterfahren zu entwickeln, welches ein anderes Clusterverfahren benötigt? Eine Antwort liegt in der Anzahl der Dimensionen. Als Anwendung wird das Verfahren für Microarray–Daten verwendet, wobei die Dimension von Datenpunkten > 7000 liegt. Durch das spektrale Clustern reduzieren sich jene auf wenige Dimensionen.

5.7. Die Zwischenbilanz

Wir haben nun gelernt, was spektrales Clustern ist. Möchte man Daten clustern, so erzeugt man zunächst einen ungerichteten, positiv gewichteten Graphen aus den Datenpunkten. Hat man die Wahl eines Clusterkriteriums getroffen, so berechnet man die entsprechende Matrix und die Eigenvektoren. Bei k Clustern verwendet man die k kleinsten oder k größten (bezüglich ihrer Eigenwerte), je nach dem Clusterkriterium. Schreibt man diese nun spaltenweise in eine Matrix, so muss man im letzten Schritt zeilenweise, mithilfe eines anderen Verfahrens, clustern.
Ab jetzt wird das Augenmerk auf die Eigenschaften des spektralen Clusterns gerichtet sein, und es wird untersucht ob und warum das Verfahren funktioniert.

5.8. Weitere Zugänge

Neben den Clusterkriterien gibt es auch noch andere Zugänge, die in diesem Kapitel erläutert werden.

5.8.1. Die Irrfahrt

Der Zusammenhang zwischen L_{sym} und L_{rw} wurde bereits gezeigt. Da L_{rw} eine Verbindung zu den Irrfahrten hat, hat man somit auch eine Verbindung von $NormalCut$ zu den Irrfahrten gefunden, die es ermöglicht, den $NormalCut$ etwas zu veranschaulichen.

Zunächst spricht man von einer Irrfahrt, wenn die Übergangswahrscheinlichkeiten alleine von den Zuständen abhängig sind. Der Zeitpunkt und die Vergangenheit (wie man in den Zustand kam) sind also irrelevant. Somit kann auch von einer homogenen Markoff–Kette (abzählbare Indexmenge) mit endlicher Zustandsmenge geredet werden. Betrachten wir einen ungerichteten, positiv gewichteten Graphen $G = (V, E, f)$, der geclustert werden soll. Wie bereits erwähnt, besitzt der transformierte gerichtete, positiv gewichtete Graph mit den Übergangswahrscheinlichkeiten $P = p_{i,j} = \frac{w_{i,j}}{d_i}$ ein Gleichgewicht. Dieses ist laut Definition invariant ($\pi P = \pi$), und wie man beweisen kann, bis auf ein multiplikatives Vielfaches eindeutig.

Satz 5.3 *Sei $P = p_{i,j} = \frac{w_{i,j}}{d_i}$ eine homogene Übergangsmatrix zu einer Markoff–Kette. Dann ist*

$$\pi := (\pi_1, ..., \pi_n) \text{ mit } \pi_i := \frac{d_i}{\mathrm{vol}(G)}$$

ein Gleichgewicht.

Beweis:
(Die Invarianz folgt aus)

$$(\pi P)_i = \sum_{j=1}^{n} p_{j,i}\pi_j = \sum_{j=1}^{n} \frac{w_{j,i}}{d_j}\frac{d_j}{\mathrm{vol}(G)} = \frac{1}{\mathrm{vol}(G)}\sum_{j=1}^{n} w_{j,i} = \frac{d_i}{\mathrm{vol}(G)} = \pi_i$$

(Die Wahrscheinlichkeitsverteilung folgt aus)

$$\pi\mathbb{1} = \sum_{i=1}^{n} \frac{d_i}{\mathrm{vol}(G)} = \frac{1}{\mathrm{vol}(G)}\sum_{i=1}^{n} d_i = \frac{1}{\mathrm{vol}(G)} * \mathrm{vol}(G) = 1$$

■

In Luxburg [13] steht folgender Satz mit Beweis:

Satz 5.4 *Sei $G = (V, E, f)$ zusammenhängend und nicht bi–partit (man kann den ungerichteten, positiv gewichteten Graphen nicht in zwei Gruppen zerlegen, so dass alle Kanten zwischen den Gruppen verlaufen). Weiterhin sei $(X_n)_{n\geq 0}$ eine Irrfahrt, wobei X_0 in seinem GGW π startet. Sind $A, B \subseteq V$ so gilt*

$$\mathrm{NormalCut}(A, \overline{A}) = \mathrm{P}(X_1 \in \overline{A} | X_0 \in A) + \mathrm{P}(X_1 \in A | X_0 \in \overline{A})$$

Beweis:
Zuerst etwas Vorbereitendes:

$$\mathrm{P}(X_0 \in A, X_1 \in B) = \sum_{i\in A, j\in B} \mathrm{P}(X_0 = i, X_1 = j) = \sum_{i\in A, j\in B} \pi_i p_{i,j} = \sum_{i\in A, j\in B} \frac{d_i}{\mathrm{vol}(V)} \frac{w_{i,j}}{d_i} = \frac{1}{\mathrm{vol}(V)} \sum_{i\in A, j\in B} w_{i,j}.$$

Benutzt man dies, kommt man zu

$$\mathrm{P}(X_1 \in B | X_0 \in A) = \frac{\mathrm{P}(X_0 \in A, X_1 \in B)}{\mathrm{P}(X_0 \in A)} = \left(\frac{1}{\mathrm{vol}(V)} \sum_{i\in A, j\in B} w_{i,j}\right) \left(\frac{\mathrm{vol}(A)}{\mathrm{vol}(V)}\right)^{-1} = = \frac{\sum_{i\in A, j\in B} w_{i,j}}{\mathrm{vol}(A)} = \frac{\mathrm{cut}(A, B)}{\mathrm{vol}(A)},$$ was ein Summand des $NormalCut$ ist.

■

Damit hat man einen intuitiven Zugang gewonnen. Man minimiert $NormalCut(A, \overline{A})$ genau dann, wenn $P(X_1 \in \overline{A} | X_0 \in A) + P(X_1 \in A | X_0 \in \overline{A})$ minimiert wird. Dies bedeutet: Die Wahrscheinlichkeit, dass der Prozess $(X_n)_n$ aus A heraus läuft, oder in A hinein läuft, am geringsten ist. Dieser Prozess $(X_n)_n$, wechselt also am unwahrscheinlichsten das Cluster (Abb. 5.5).
Einer der schönsten Sätze, bezüglich der Markoff–Ketten, ist für mich der Birkhoffsche Ergodensatz (s. Ritter [20]), der eine Verbindung zwischen den durchschnittlichen Besucheranzahlen der Zustände und dem Gleichgewicht herstellt. Meine Intention wäre, die Cluster in einem Zustand zusammenzufassen, und zwischen den Clustern einen neuen Zustand einzuführen, um dadurch etwas über die relative Häufigkeit des Clusterwechsels auszusagen (Abb. 5.6). Leider ist dieser neue Prozess nur in Spezialfällen (z.B. die Ähnlichkeitsmatrix hat lauter Einsen) eine Markoff–Kette (s. Ritter [20] *lumpability*). Somit ist es nicht möglich, eine Aussage über die relative Häufigkeit des Clusterwechsels zu treffen.

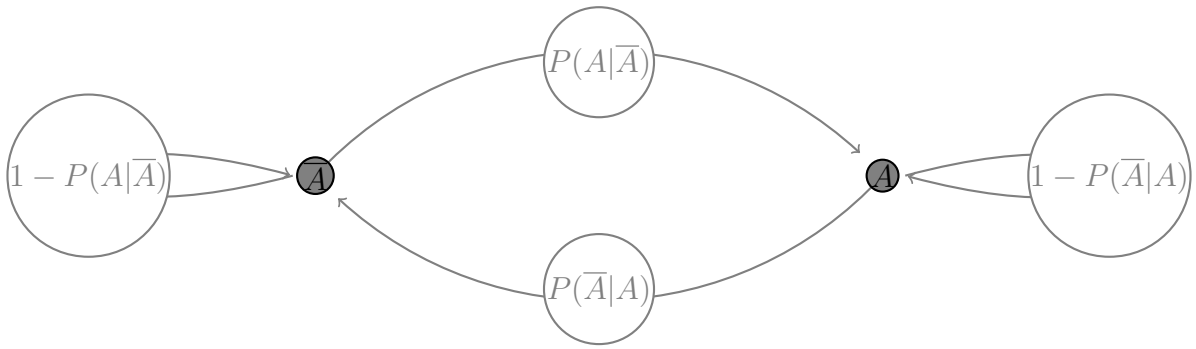

Abbildung 5.5.: Vereinfacht dargestellte Irrfahrt

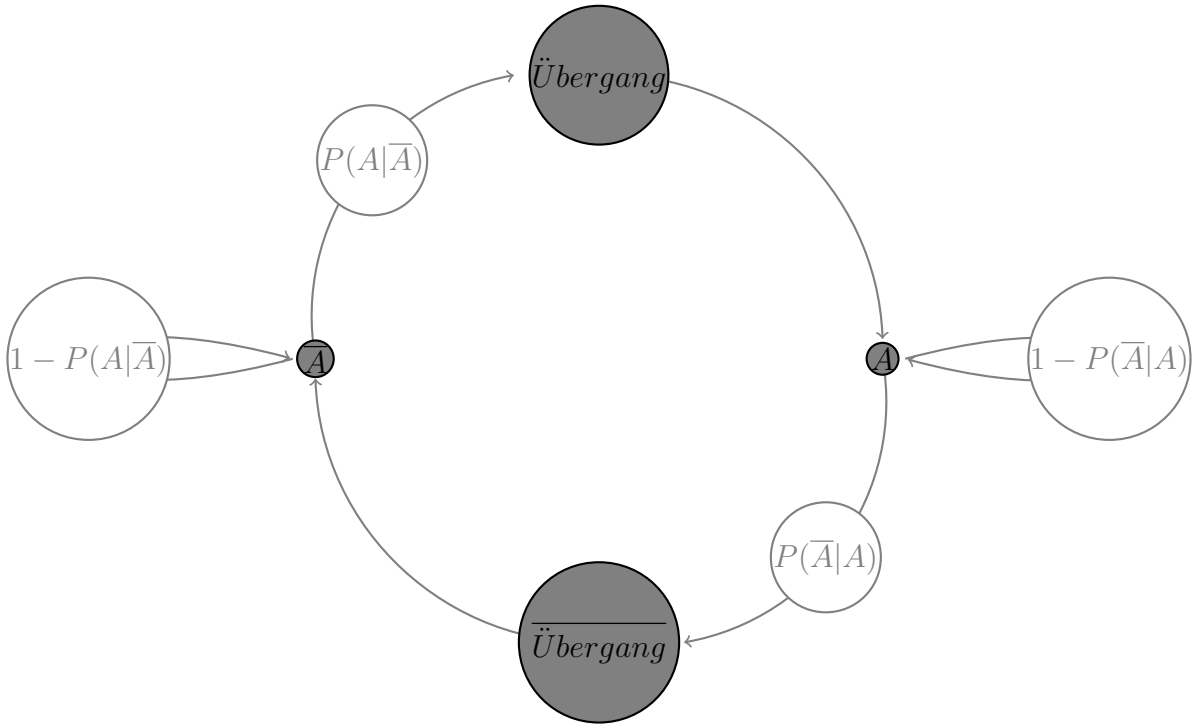

Abbildung 5.6.: Gruppierte Irrfahrt mit zwei neuen Zuständen

5.8.2. Die Transposition

Eine Anschauung für den *NormalCut* wurde bereits dargelegt. Nun wird eine weitere Anschauung betrachtet, welche für alle drei Clusterkriterien anwendbar ist. Begonnen wird mit dem optimalen Fall (unzusammenhängender ungerichteter, positiv gewichteter Graph), indem die Matrix

$$L = \begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{bmatrix}$$

betrachtet wird. Stört man nun diese Matrix

$$\widehat{L} = L + \delta L,$$

wobei δL eine Matrix ist, die L nicht signifikant stört und ausschließlich Werte nahe 0 hat. So hat diese kleine Störung keinen Einfluss auf die Partition. Sei L_{trans} eine durch Transpositionen entstandene Matrix von $\widehat{L}$, die geclustert werden soll. Das menschliche Auge wird nicht mehr in der Lage sein, diese Matrix durch bloßes Betrachten zu gruppieren. Man könnte das Clustern als eine Methode ansehen, bei der man eine Matrix in die ähnliche Gestalt einer diagonalen Blockmatrix überführt. Durch eine Partition könnte man so L_{trans} wieder zu $\widehat{L}$ umformen.

5.9. Die Eigenvektoren im Optimalfall

Wie schon erwähnt, sollten die Eigenvektoren ähnlich den δ_i sein. Im Optimalfall hat man einen ungerichteten, positiv gewichteten Graphen vor sich, der in k Zusammenhangskomponenten zerfällt (es gibt keine Ähnlichkeit zwischen den Clustern). Ist dies erfüllt, so entsprechen die kleinsten k Eigenvektoren tatsächlich den Dirac Vektoren. Zumindest bei *RatioCut* und *NormalCut*.

5.9.1. Der RatioCut

In Luxburg [13] findet man folgenden Satz:

Satz 5.5 *Sei $G = (V, E, f)$ ein ungerichteter Graph mit nicht negativen Kantengewichten, dann stimmt die Vielfachheit k des Eigenwertes 0 mit der Anzahl der Zusammenhangskomponenten $(A_1, ..., A_k)$ überein. Der Eigenraum des Eigenwertes 0, wird durch die Dirac Vektoren δ_i (hier betrachtet man Zusammenhangskomponenten und nicht Cluster), aufgespannt.*

Beweis:
(Induktion über) Betrachtet man k Zusammenhangskomponenten.

(Induktionsbeginn) Gestartet wird mit $k = 1$ (der ganze Graph ist zusammenhängend).

(Die Eindeutigkeit folgt aus) Nimmt man an f sei ein Eigenvektor mit Eigenwert 0. Es gilt nun mit Hilfe von Satz 5.1

$$0 = f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2.$$

Da die Summanden > 0 sind, müssen alle Summanden $w_{i,j}(f_i - f_j)^2 = 0$ sein. Ist $w_{i,j} > 0$, so muss $f_i = f_j$ sein und man kann daraus schließen, dass f konstant ist.

(Die Gleichheit folgt aus) Ist nun der Graph zusammenhängend, so ist $\mathbb{1}$ Eigenvektor zum Eigenwert gleich 0. Dies entspricht dem Dirac Vektor.

(Der Induktionsschritt) Sei nun k beliebig wählbar.
O.B.d.A. seien die Objekte ihren Zusammenhangskomponenten zugeteilt, so dass die Matrix

$$L = \begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{bmatrix}$$

in diagonaler Blockform ist. Dabei sind L_i eigenständige Laplacematrizen, zu ihren zugehörigen Zusammenhangskomponenten.
Das Spektrum von L soll berechnet werden. Es ist anzumerken, dass L eine diagonale Blockmatrix ist, weshalb man das Spektrum durch die Vereinigung aller Spektren der L_i erhält. Zu beachten ist, dass dabei die Eigenvektoren der L_i, entsprechend mit 0 aufgefüllt werden.
Wir wissen aus Satz 5.1, dass ein kleinster Eigenvektor (bezüglich der Eigenwerte) der konstante $\mathbb{1}$ Vektor ist. Damit entspricht dieser Eigenvektor bezüglich L_i dem Dirac Vektor δ_i, bezüglich L. Da k Zusammenhangskomponenten betrachtet werden, hat man auch k verschiedene δ_i Eigenvektoren gefunden, die aufeinander senkrecht stehen und somit den k dimensionalen Eigenraum des Eigenwertes 0 von L aufspannen.

■

Berechnet man die k kleinsten Eigenvektoren (bezüglich der Eigenwerte) von L, so tragen diese tatsächlich die Clusterinformationen in sich. Daher weiss man, dass bei optimalen Fällen, spektrales Clustern funktioniert.

5.9.2. Der NormalCut

Benutzt man Satz 5.2 und $D^{1/2}\mathbb{1}$, anstelle von Satz 5.1 und $\mathbb{1}$, so kommt man zu dem gleichen Ergebnis, wie bei *RatioCut*.

5.9.3. Der MinMaxCut

Da $\widehat{W}$ nicht wie L, L_{sym} positiv semi–definit ist, kann man zumindest die Beweisführung nicht verallgemeinern. Eine vergleichbare Aussage konnte bisweilen nicht getroffen werden.

5.10. Die Perturbationstheorie

Bisher haben wir gelernt, was spektrales Clustern ist und anschaulich dargestellt. Dass spektrales Clustern bei einem optimalen Fall funktioniert, wurde für den *RatioCut* und den *NormalCut* gezeigt. Wie sieht es jedoch bei einem nicht optimalen Fall aus? Um diese Frage zu beantworten, wird wieder der optimale Fall betrachtet und man wirkt etwas störend auf ihn ein. Man wird sehen, dass es eine Abschätzung gibt, die etwas über die Güte von den Eigenvektoren aussagt.
Die Perturbationstheorie ist die Grundlage dafür, dass spektrales Clustern bei nicht optimalen Fällen (die Ähnlichkeit zwischen Clustern ist nicht 0) funktioniert. Man geht von einem optimalen Fall mit Matrix L aus und stört ihn durch eine Störmatrix H, was zu der nicht optimalen Matrix $\widehat{L} := L + H$ führt.
Zur Vorbereitung:

Definition 5.10.1 *Sei die Matrix $A \in \mathbb{R}^{m\times n}$, dann ist $||A||_F$*

$$||A||_F := \sqrt{\text{Tr}(AA^T)},$$

die Frobenius Norm der Matrix A, wobei Tr *die Spur einer Matrix ist.*

Seien $V_1, V_2 \in \mathbb{R}^{n \times k}$, wobei die k Spalten jeweils in $\mathbb{R}^n$ orthonormal sind und $0 \leq \theta_1 \leq ... \leq \theta_k \leq \frac{\pi}{2}$, die kleinsten Winkel zwischen entsprechenden Basisvektoren darstellen. Anschaulich gesprochen entspricht jede Spalte (Vektor) in V_1 einer Spalte (Vektor) in V_2, wobei diese zwei Vektoren, solange sie nicht gleich sind, eine Ebene aufspannen und sich in dieser Ebene um einen Winkel unterscheiden. Sind Sie gleich, so entspricht der Winkel 0. Notationell stelle $\sin\Theta(V_1, V_2)$ eine Diagonalmatrix mit der Diagonalen $\sin\Theta(V_1, V_2)(i, i) = \sin\theta_i$ dar. Dann gilt folgender Satz (s. Luxburg [13]):

Satz 5.10.1 *Seien $A, H \in \mathbb{R}^{n \times n}$ symmetrische Matrizen, und $||\cdot||$ die Frobenius Norm, oder die Zweier–Norm für Matrizen. Betrachtet man $\widehat{A} := A + H$ als eine gestörte Matrix von A und sei $S_1 \subset \mathbb{R}$ ein Intervall. Unter $\sigma_{S_1}(A)$ versteht man die Menge aller Eigenwerte die innerhalb von S_1 liegen, und unter V_1 den entsprechenden Eigenraum. Für $\sigma_{S_1}(\widehat{A})$ und $\widehat{V}_1$ sei es genauso definiert nur für $\widehat{A}$. Definiert man den Abstand zwischen S_1 und dem Spektrum ausserhalb von S_1 durch*

$$\delta = \min\{|\lambda - s|; \lambda \text{ Eigenwert von } A, \lambda \notin S_1, s \in S_1\},$$

so ist die Distanz $d(V_1, \widehat{V}_1) := ||\sin\Theta(V_1, \widehat{V}_1)||$ zwischen den zwei Unterräumen V_1 und $\widehat{V}_1$ nach oben durch

$$\mathrm{d}(V_1, \widehat{V}_1) \leq \frac{||H||}{\delta}$$

beschränkt.

Betrachten wir diesen Satz an einem Beispiel. Gestartet wird wieder mit dem Optimalfall (k Zusammenhangskomponenten), mit den Dirac Vektoren als Eigenvektoren von L_{sym} oder L. Sei weiterhin der spektrale Sprung $|\lambda_k - \lambda_{k+1}|$ relativ groß. Stört man L oder L_{sym} nur gering, mit der Matrix H so ist $||H||_F$ und auch $\frac{||H||_F}{|\lambda_k - \lambda_{k+1}|}$ relativ klein. Sei $\widehat{L} = L + H$ und $\widehat{L}_{\mathrm{sym}} = L_{\mathrm{sym}} + H$, so ist auch $d(L, \widehat{L})$, oder $d(L_{\mathrm{sym}}, \widehat{L}_{\mathrm{sym}})$ relativ klein, was auf kleine Winkel zwischen den Basisvektoren der Orthonormalbasen schließen lässt. Anders ausgedrückt haben sich die Dirac Vektoren nur ein wenig gedreht.
Ist jedoch die Störung H zu groß, so sind die Eigenvektoren unterschiedlicher zu den Dirac Vektoren und verlieren das Wissen über die richtige Partition.
Zusammenfassend für k Cluster ist die Wahl von $S_1 = [0, \lambda_k]$ und der spektrale Sprung $|\lambda_k - \lambda_{k+1}|$ wichtig. Umso größer der Sprung und umso kleiner die Störung H ist, desto ähnlicher sind die Eigenvektoren. Somit tragen auch die gestörten Eigenvektoren noch die richtige Partition in sich.
Dies ist die Begründung dafür, warum spektrales Clustern überhaupt funktionieren kann.

5.11. Die Komplexitätsabschätzung

Wie schon einmal erwähnt, ist es NP–hart eine optimale Partition bezüglich eines Clusterkriteriums zu berechnen. Wie sieht dies bei dem spektralen Clustern aus? Hier kann leider kein scharfer Wert gegeben werden, denn es kommt darauf an, welchen Algorithmus man verwendet, um aus den Eigenvektoren eine Partition zu berechnen. Lässt man dies außen vor, so reduziert sich das spektrale Clustern auf die Berechnung der k ersten Eigenvektoren, was im Allgemeinen in $O(n^3)$ (bei n Datenpunkten) geschieht (s. Press et al. [16]).

5.12. Die Fehlerabschätzung

Mit Hilfe von Clusterkriterien kann man eine Partition bewerten. Es wäre erstrebenswert, dass man durch spektrales Clustern in der Lage wäre, Partitionen zu erzeugen, die maximal um ein x–faches schlechter sind als eine optimale Partition. Hat zum Beispiel eine optimale Partition bezüglich eines Clusterkriteriums den Wert 3, so hätte man gerne eine Abschätzung, die garantiert, dass spektrales Clustern maximal eine Partition mit einem Wert von $1,5 * 3$ bezüglich desselben Clusterkriteriums generiert. Oft wird dieser Sachverhalt durch ein Verhältnis, wie zum Beispiel 16 zu 9 oder 4 zu 3, ausgedrückt.
Hätte man eine spektrale Clusterung $B_1, ..., B_k$ eines ungerichteten, positiv gewichteten Graphen $G = (V, E, f)$ und eine exakte $A_1, ..., A_k$ so hätte man gerne, wie bereits erwähnt, eine Abschätzung über das Verhältnis von zum Beispiel $RatioCut(B_1, ..., B_k)$ zu $RatioCut(A_1, ..., A_k)$, um etwas über die Güte des spektralen Clusterns aussagen zu können. Leider gibt es ein Gegenbeispiel, welches keine Abschätzung zulässt. Es handelt sich dabei um den cockroach Graphen (s. Luxburg [13]), der wie eine Leiter mit fehlenden Sprossen aussieht und als Gewichtsfunktion die konstante Funktion $\mathbb{1}$ hat (Abb. 5.7).

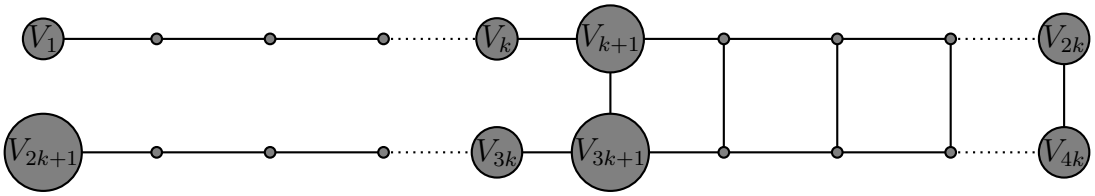

Abbildung 5.7.: cockroach Graph aus Luxburg [13]

Erinnert man sich an den *RatioCut*, so bevorzugt dieser gleich große Cluster, falls sich der *cut* dabei nicht groß verändert. Er würde den Graphen in der Mitte vertikal durchtrennen und somit $A_1 = \{V_1, ..., V_k, V_{2k+1}, ..., V_{3k}\}, A_2 = V \backslash A_1$ als Resultat haben. Jedoch schneidet der spektrale Algorithmus den cockroach Graphen horizontal

durch (s. Luxburg [13]) und gibt somit $B_1 = \{V_1, ..., V_{2k}\}, B_2 = V \backslash B_1$ zurück. Nun ist $RatioCut(A_1, A_2) = \frac{2}{2k} + \frac{2}{2k} = \frac{2}{k}$, wobei $RatioCut(B_1, B_2) = \frac{k}{2k} + \frac{k}{2k} = 1$ ist. Da man k beliebig wählen kann, ist auch das Verhältnis beliebig schlecht.
Der gleiche Fall kann für $NormalCut$ durchgeführt werden (s. Luxburg [13]).

5.13. Exakt versus spektral

Nun zu einigen Fallbeispielen. Wie man bei kleinem n effizient eine optimale Partition berechnet, wurde bereits erläutert. Dadurch konnten einige Fallbeispiele berechnet werden, die jetzt mit den Ergebnissen des spektralen Clusterns verglichen werden.

5.13.1. Der euklidische Abstand als Ähnlichkeit

Wir haben bereits ausführlich über die Merkmalstransformation gesprochen. Hier wird diese Transformation nicht zu ausführlich benutzt, da etwas über die geometrischen Eigenschaften (wie clustern die Kriterien) in Erfahrung gebracht werden soll. Deshalb wird der euklidische Abstand in eine Ähnlichkeit transformiert, indem der Abstand von dem maximalen Abstand subtrahiert wird ($s(x, y) = \max_{i,j} d_{i,j} - d_{x,y}$, wobei $d_{x,y}$ der euklidische Abstand von Punkt x zu y ist). Haben innerhalb eines Datensatzes zwei Punkte maximal einen Abstand von 5, so haben zwei Punkte, die einen Abstand von 3 haben, eine Ähnlichkeit von $5 - 3 = 2$.

5.13.2. Der graphentheoretische Zugang

Zunächst berücksichtigt der $RatioCut$ die selbstreferenzierenden Kanten nicht, da er nur den cut betrachtet. Der $NormalCut$ und der $MinMaxCut$ benutzen die selbstreferenzierenden Kanten jedoch durchaus. Der $NormalCut$ hat das vol, der $MinMaxCut$ den cut im Nenner stehen. Wie bereits gezeigt, gilt $vol(A_1) = cut(A_1, A_1) + cut(A_1, \overline{A})$. Setzt man jetzt die Selbstähnlichkeit sehr hoch an, so werden sich $NormalCut$ und $MinMaxCut$ ähnlicher, da der $cut(A_1, \overline{A})$ verhältnismäßig kleiner wird. Deshalb wird immer ein ungerichteter, positiv gewichteter Graph ohne selbstreferenzierende Kanten betrachtet, da dieses Vorgehen eine gewisse Normalisierung darstellt.

5.13.3. Der Vergleich

Als Vergleichsdatensatz benötigt man einen Datensatz, der durch alle drei Clusterkriterien gut (nicht unbedingt perfekt) geclustert wird. Er sollte jedoch nicht trivial sein,

weshalb ich mich für einen sphärisch gleich verteilten Fall mit unterschiedlicher Anzahl von Datenpunkten entschieden habe. Würde man eine Niveaulinie einzeichnen, so würden beide Cluster einen gleich großen Kreis darstellen. In Abbildung 3.8 und Abbildung 3.7 (Kapitel 3) sehen Sie die exakten Partitionen. Das Ergebnis des $MinMaxCut$ entspricht dem des $NormalCut$, weshalb hier auf eine weitere Darstellung verzichtet wird. In Abbildung 5.8 sind die Ergebnisse des spektralen Clusterns zu sehen. Sie stimmen alle mit ihren exakten Lösungen überein. Schön zu sehen sind auch die 1–Simplexe (Geraden) der Eigenvektoren, wie sie im $\mathbb{R}^2$ liegen.

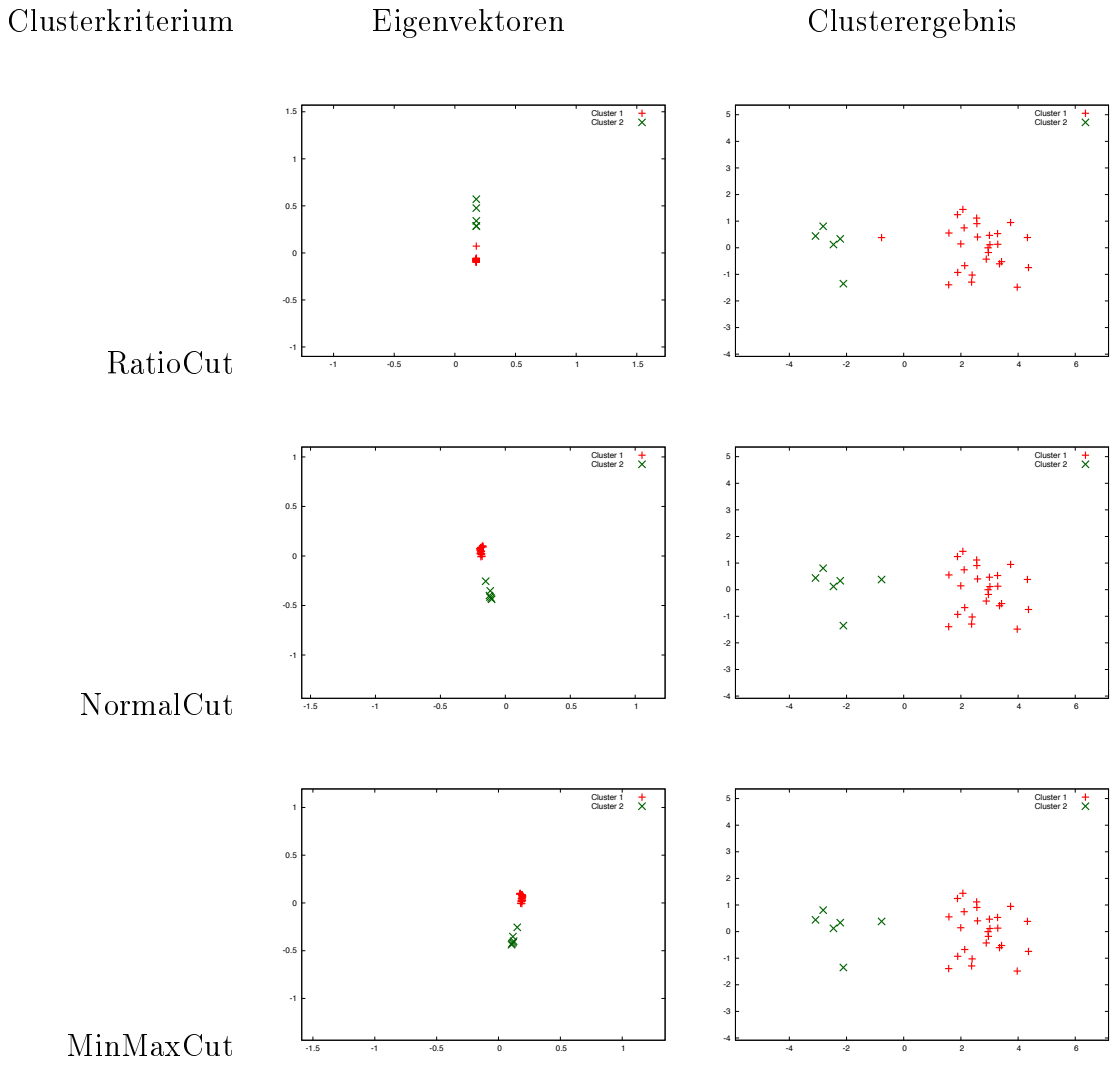

Abbildung 5.8.: Spektrale Ergebnisse mit ihren Eigenvektoren

5.14. Die Zusammenfassung

Zunächst ein kurzer Rückblick. Die Clusterkriterien wurden mit Hilfe von Dirac Vektoren in eine Summe von Rayleigh–Quotienten umgeformt. Durch die Relaxation der Bedingungen (das Aussehen) an die Dirac Vektoren, konnte man die Minimierungsprobleme der Clusterkriterien auf die Eigenvektoren zurückführen. Wendet man nun einen anderen Clusteralgorithmus auf diese Vektoren an, so bekommt man eine Näherung des Ergebnisses. Die Auswahl eines bestimmten Clusteralgorithmus ist mit Vorsicht zu treffen, denn alle haben Vor– und Nachteile. In meiner Implementierung habe ich *k–means* und MST verwendet. *k–means* ist ein sehr berühmter Clusteralgorithmus, der gleich große Cluster bevorzugt und gegenüber Ausreißern toleranter als MST ist. Der MST hat seinen größten Vorteil in der Laufzeit und seinem Nicht–Beachten der Clustergröße. Auf die Resultate von MST und *k–means* kann man das gewählte Clusterkriterium anwenden und das bessere Ergebnis selektieren. Leider gibt es hier keine Verallgemeinerung welches Clusterkriterium immer am Besten funktioniert. Clustert man so zum Beispiel die *Golub et al. [8]* Daten, die später noch genauer betrachtet werden, so gewinnt bei zwei Clustern MST, und bei drei Clustern *k–means*.
Bei dem spektralen Clustern handelt es sich um eine Approximation und somit nicht um eine exakte Lösung. Man hofft natürlich, dass dieses Ergebnis nahe dem Optimum liegt. Durch eine schrittweise Neuzuordnung kann man, diese Approximation weiter verbessern. So ordnet man denjenigen Punkt neu zu, der den Clusterkriteriumswert am meisten verbessert. Dieses Vorgehen muss nicht in einem globalen Optimum landen, jedoch kann man so ein lokales Optimum erreichen (greedy–Algorithmus). Allgemein eignet sich die Partition, die spektrales Clustern liefert, als Startwert für feinere iterative Optimierungsalgorithmen.
Man könnte das spektrale Clustern auch als eine erweiterte Merkmalstransformation sehen, die auf einen $\mathbb{R}^k$ abbildet.

Kapitel 6.

Die Anzahlbestimmung der Cluster

Die Anzahl der Cluster ist meist unbekannt und soll jetzt bestimmt werden. Dies ist jedoch im Allgemeinen sehr schwer und es gibt viele verschiedene Vorgehensweisen dazu. Hier wird auf zwei Verfahren von allgemeiner Gültigkeit aufmerksam gemacht und auf ein weiteres, welches sehr speziell ist.
Die Komplexität dieses Themenabschnittes würde den Rahmen dieser Diplomarbeit sprängen, weshalb hier nur kurz darauf eingegangen wird.

6.1. Die allgemeinen Verfahren

Das erste Verfahren, welches betrachtet wird, ist am intuitivsten und wird auch später angewandt. Das Zweite ist dagegen ein sehr bekanntes Verfahren und sagt etwas über die Stabilität des Ergebnisses aus.

6.1.1. Das natürliche Clusterverhalten

In verschiedenen Arbeiten liest man über einen 2–Clusterfall, welcher zu einem Mehrclusterfall verallgemeinert wird. Dabei werden die Cluster des 2–Clusterfalls auf weitere Partitionen untersucht, und so weiter. Man hat somit einen divide and conquer Algorithmus, der eine Partition baumartig generiert. Im Gegensatz dazu gibt es viele Algorithmen die von Anfang an mit mehreren Clustern starten (der k–Clusterfall ist unabhängig von dem Ergebnis des $k-1$ Clusterfalls). Bei dem divide and conquer Prinzip ist immer eine echte Teilmengenbeziehung zu **einem** anderen Cluster gewahrt (außer die Grundmenge selbst), wohingegen dies für den anderen Ansatz nicht gilt. Man kann jetzt diese Teilmengenbeziehung nutzen um die Clusteranzahl zu bestimmen.
Betrachtet man drei Cluster, wobei zwei von diesen Clustern sich sehr ähnlich sind. Clustert man diese drei Cluster in zwei Cluster, so würde man von einem guten Clusterkriterium erwarten, dass es die zwei ähnlicheren zusammenfasst (Abb. 6.1). Clustert

man diese Daten erneut, aber diesmal in drei Cluster, so würde man feststellen, dass eins der Cluster aus der ersten Partition, in zwei Cluster zerfallen ist. Das Ganze könnte man so lange wiederholen bis man feststellt, dass kein Cluster zerfallen wird, sondern dass das neue Cluster aus mehreren Clustern entstanden sein wird (verletzt die Teilmengenbeziehung, s. Abb. 6.2).
Diese Idee ist sicherlich nicht immer richtig, jedoch für ein breites Spektrum geeignet.

Abbildung 6.1.: Drei Cluster wobei zwei ähnlich zueinander sind (Clusterübergreifende Kanten weggelassen).

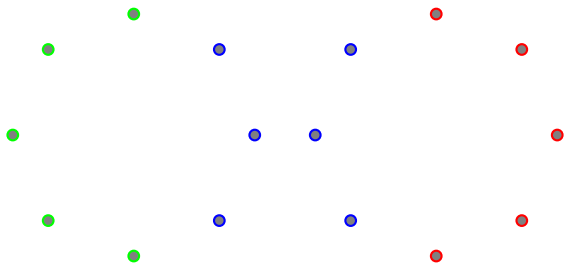

Abbildung 6.2.: Zwei Cluster die in drei Cluster unterteilt werden (Kanten weggelassen). Jede Farbe entspricht einem Cluster.

6.1.2. Die Stabilität (Weglassen von Daten)

Die zweite Vorgehensweise wäre es, einige Datenpunkte wegzulassen. Dies bewirkt, dass sich bei einer stabilen Partition nicht sehr viel ändert. Jedoch bewirkt diese Vorgehens-

weise bei einer unstabilen (falschen) Partition eine stärkere Abweichung der Resultate. Dieses Verhalten kann durch das natürliche Clusterverhalten erklärt werden. Betrachtet man wieder Abbildung 6.1 und denkt sich ein paar Datenpunkte weniger, so sollte man noch immer die zwei Cluster entdecken können. Das Weglassen von Datenpunkten hat keinen Einfluss auf die übrigen Zuordnungen. Betrachtet man jedoch Abbildung 6.2, lässt hier ein paar Datenpunkte weg und versucht gleich große Cluster zu erzeugen, so ändern sich die Zuordnungen durchaus. Als Beispiel lassen wir die zwei oberen blauen Datenpunkte weg, dadurch reduziert sich das blaue Cluster auf vier Datenpunkte und holt sich aus dem roten, oder dem grünen Cluster zumindest einen Datenpunkt. Dieser Datenpunkt ist also neu zugeordnet worden und deutet somit auf eine zu große Clusteranzahl hin. In verschiedenen arbeiten kennt man dieses Verfahren auch unter Kreuzvalidierungsverfahren (s. Sick [25]).

6.2. Die Eigenwerte

Wir hatten bereits bei der Perturbationstheorie über den Eigensprung $|\lambda_k - \lambda_{k+1}|$ gesprochen. Bei dem betrachteten Optimalfall (unzusammenhängender ungerichteter, positiv gewichteter Graph) sind, wie bereits bewiesen, die ersten k Eigenwerte 0. Tatsächlich kann man einen ungewöhnlich hohen Sprung dazu nutzen, um eine Clusteranzahl zu bestimmen (s. Luxburg [13]). Verwischen jedoch die Clustergrenzen zu sehr, so verschwindet auch der Sprung in den Eigenwerten. Dies ist letzten Endes nur konsequent, da es keine klare Partition gibt (Abb. 6.3). Bei einer genaueren Betrachtung der Beispiele in Luxburg [13], ist dieses Verfahren nur bedingt einsetzbar. Betrachtet man ein Beispiel mit vier Clustern, so ist zwar ein spektraler Sprung bei λ_4 zu λ_5 zu sehen, jedoch auch oft von λ_8 zu λ_9 (Abb. 6.3).

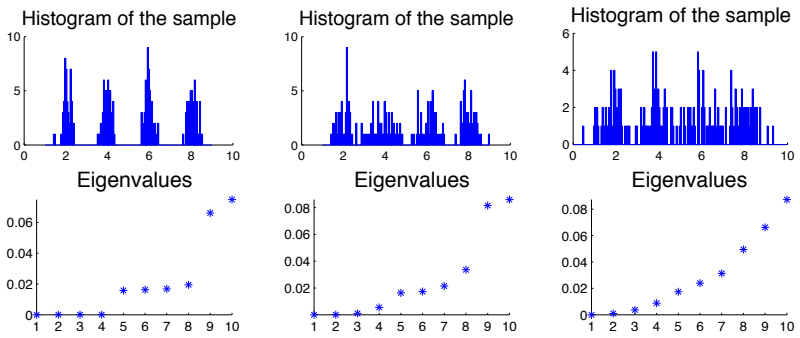

Abbildung 6.3.: (s. Luxburg [13]) Drei Datensätze und die kleinsten 10 Eigenwerte bezüglich L_{rw}.

Kapitel 7.

Die Microarray–Daten

In diesem Kapitel wird eine praktische Anwendung betrachtet. Dabei konzentrieren wir uns auf Microarray–Daten. Im Fachbereich Biologie sind sie vielseitig einsetzbar und haben ein großes Potenzial für die Erforschung von Krankheiten.
Zunächst werden grundlegende Begriffe eingeführt, wird erklärt was Microarray–Daten sind und wozu man sie benötigt. Anschließend wird eine Ähnlichkeit auf Microarray–Daten definiert, um die Merkmalstransformation durchführen zu können. Als Anwendung werden einige Datensätze mit Hilfe von spektralem Clustern untersucht. Da nur Datensätze betrachtet werden, von denen die Partitionen bereits bekannt sind, wird es anschließend möglich sein, eine Bewertung abzugeben. Wichtig dabei ist, dass dieses Wissen über die richtige Partition nicht in die Merkmalstransformation mit einfließt. Um eine genauere Merkmalstransformation treffen zu können, wird eine alternierende Erweiterung aufgezeigt, was die Ergebnisse nochmals verbessern wird. Die Ergebnisse, die mit Hilfe der von mir gewählten Merkmalstransformation erzielt werden, werden zum Schluss mit anderen Arbeiten verglichen.

7.1. Die Grundlagen über DNA und RNA (Merkmale)

Desoxyribonukleinsäure (DNA, DNS) ist ein Biomolekül, welches in jedem Lebewesen vorkommt und oft als Doppelhelix dargestellt wird. Die Ribonukleinsäure (RNA, RNS) hingegen ist eine Übersetzung der DNA, wobei T durch U ersetzt wird und die RNA im Regelfall nur aus einem Strang besteht. Der Prozess der Gewinnung der RNA aus der DNA wird Transkription genannt, und ist bei jeder Zelle unterschiedlich. Dieser Unterschied besteht in dem Prozess der Genregulation, also der Steuerung der Genexpression (eine Steuerung der Aktivität von Genen).

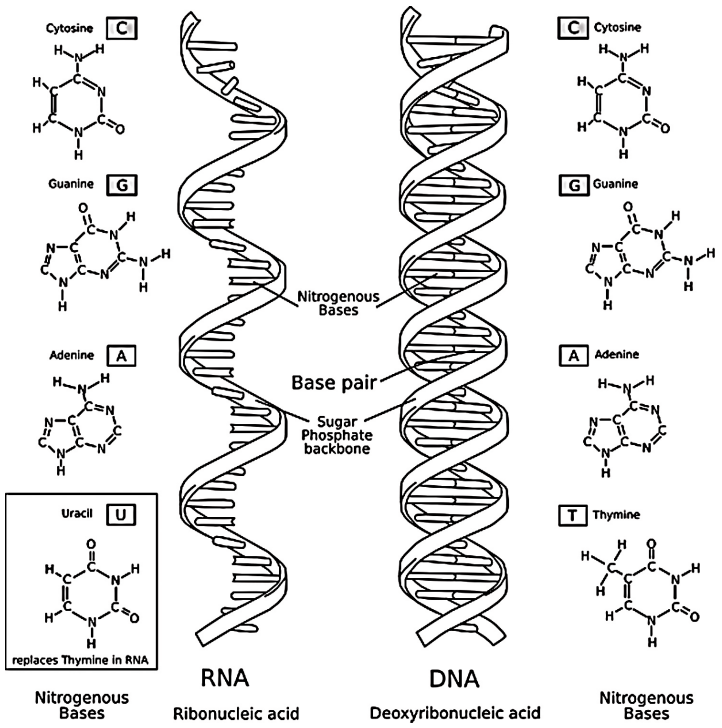

Abbildung 7.1.: DNA vs. RNA [2]

7.2. Was sind Microarrays

Microarrays benutzen die RNA um Rückschlüsse auf aktive Gene zu erhalten. Bei diesen Microarrays handelt es sich um kleine Chips, die in viele kleine Bereiche unterteilt sind, welche Indikatoren (Probes) für ein bestimmtes Merkmal (spezifisches Gen, Target) sind. Bevor man Zellgewebe mit der RNA auf den Chip aufträgt (besteht aus vielen Targets die sich an den Probes binden), werden Farbstoffe beigemischt, womit man später die RNA Bestandteile (die Targets) sichtbar machen wird. Jeder einzelne Bereich des Chips ist in der Lage ein Merkmal (Gen) an sich zu binden (Target – Probe Bindung). Mit Hilfe eines Lasers kann man den Chip abtasten und die Verfärbungen protokollieren. Je nach Verfärbung eines Bereiches, kann man Rückschlüsse auf das Expressionsprofil (entspricht einer konkreten Genexpression) ziehen. Die Anzahl der Merkmale, beziehungsweise der Bereiche, variieren pro Chip. Es werden ausschließlich Daten mit mehr als 7000 Merkmalen betrachtet.
Zu unterscheiden bei den Microarrays sind die kommerziellen, wie zum Beispiel Affymetrix GenChips® und die freien, die selbsterstellten Microarrays. Bei den von uns betrachteten Anwendungsdaten (s. Golub et al. [8] und Stegmaier et al. [26]) handelt es sich immer um Daten die von den GenChips® von Affymetrix stammen.

7.3. Wozu sind Microarrays gut

Hier gibt es viele Möglichkeiten von Anwendungen. So kann man durch die Analyse der Daten Rückschlüsse auf Krankheiten, Medikamentenunverträglichkeiten, auf den Erfolg einer Behandlung, und so weiter, schließen. In diesem Fall soll herausgefunden werden, ob eine bestimmte Krankheit untergliedert werden kann (wie z.B. Krebs in Blut– und Knochenkrebs), um somit eine gezieltere Behandlung zu ermöglichen. Es geht hierbei jedoch mehr um die Beurteilung der Partition der Testdaten, weshalb eine Krankheit gewählt wird, die bereits gut erforscht ist.

7.4. Das spektrale Clustern von Microarray–Daten (Ähnlichkeiten)

Das Ziel ist es, Patienten zu clustern, weshalb zunächst aus den Merkmalen eine Ähnlichkeit gewonnen werden muss. Dies ist nicht so leicht, da viele Merkmale vorhanden sind, die mit der betrachteten Krankheit in keinem Zusammenhang stehen. Diese überflüssigen Merkmale werden ein gewisses Rauschen verursachen und dafür sorgen, dass nicht immer ein perfektes Ergebnis entstehen kann. Diese Unbekannte macht die Wahl einer sinnvollen Transformation sehr schwer. Nach einiger Recherche ist mir Qiu und Plevritis [17] aufgefallen. Dort wird als Ähnlichkeit

$$w_{i,j} = \left(\frac{1}{1 + d_{i,j} - min_{u,v} d_{u,v}} \right)^k$$

angegeben, wobei $d_{i,j}$ der euklidische Abstand von Punkt i zu Punkt j ist und $k \in \mathbb{R}_+$. Hierbei handelt es sich um eine schnell fallende Funktion, die im Nullpunkt gleich 1 ist, und entfernt sich ein Punkt immer weiter, gegen 0 geht (Abb. 7.2). Mit Hilfe dieser Funktion klassifiziert und clustert Qiu und Plevritis [17] einige Daten recht erfolgreich. Anzumerken ist, dass es sich bei dem von Qiu und Plevritis vorgestellten Algorithmus um eine überwachte Klassifikation und Clusterung handelt. „Überwacht“ impliziert dabei das Wissen über die richtige Partition. Man stellt den Parameter k so ein (händische Regelbestimmung), dass es bei dem Clustern zu den erwünschten Resultaten kommt (man trainiert k) und erhält dadurch eine Klassifikation. Die so gewonnenen Resultate werden dann auf weitere Cluster untersucht, indem man die einzelnen Cluster weiter clustert (divide and conquer). Zusammengefasst betreiben sie überwachte Klassifikation mit anschließendem clustern. Bei dem Datensatz Golub et al. [8] verwenden Qiu und Plevritis weiterhin nur die Aussagestärksten 1000 Gene (die am unterschiedlichsten ausgeprägten Gene in den zwei bekannten Clustern), was bei einer überwachten Klassifikation durchaus erlaubt ist. Bei dem anschließenden clustern ist dies jedoch kontraproduktiv,

da Merkmale, die gerade ein zusätzliches Cluster beschreiben, bei der Selektion verloren gehen können.

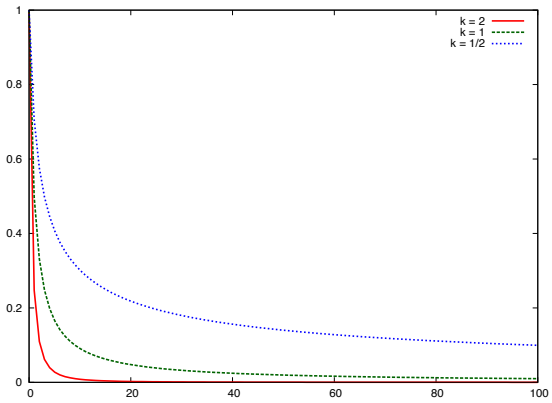

Abbildung 7.2.: $w(x) = \left(\frac{1}{1+x}\right)^k$

Nichts desto trotz war diese Arbeit der Ideengeber dafür, eine ähnliche Funktion zu verwenden, welche keinen Parameter hat, oder sich abschätzen lässt. Aus Ritter [21] ist mir unter anderem die Gaußsche Glockenkurve in Erinnerung geblieben. Die Schlüsseleigenschaft der Funktion in Qiu und Plevritis [17] ist, dass entfernte Objekte viel unähnlicher sind als Objekte die näher beieinander liegen. Dies gilt auch für die Gaußsche Glockenkurve, die noch einen weiteren Vorteil hat: So nimmt die Kurve anfangs nicht so schnell wie bei Qiu und Plevritis [17] ab (vgl. Abb. 7.3 und Abb. 7.2). Dies sorgt dafür, dass nahe beieinander liegende Objekte auch ähnlich bleiben.

7.4.1. Die Gaußsche Glockenkurve

Die Gaußsche Glockenkurve ist weit verbreitet und findet viele Anwendungen.

Definition 7.1 *(s. Röblitz und Weber [22]) Die Gaußsche Ähnlichkeit ist definiert durch*

$$s_{i,j} := \exp(-\beta d_{i,j}^2) \text{ wobei } \beta = \frac{1}{2\sigma^2},$$

$d_{i,j}$ ist der euklidische Abstand zwischen Punkt i und j und $\sigma \in \mathbb{R}$ ein Parameter, der die Nachbarschaft kontrolliert. In Abbildung 7.3 ist diese Funktion dargestellt. Sie hat einen wichtigen Vorteil gegenüber der Funktion, die in Qiu und Plevritis [17] vorgestellt wird. Man kann den Parameter σ schätzen. Nimmt man an, man hätte zwei Cluster vor

sich, so berechnet man den durchschnittlichen Abstand x_{ave} aller Punkte voneinander. Anschaulich sollte hier in etwa die halbe Strecke von einer Clustermitte zur anderen herauskommen. Sind zwei Datenpunkte weiter als x_{ave} voneinander entfernt, so sollte die Ähnlichkeit $< 0,5$ sein, da die Ähnlichkeiten zwischen 0 und 1 liegen. Die geschwungene Form der Gaußschen Ähnlichkeit, macht ähnliche Punkte ($d_{i,j} < x_{\text{ave}}$) schneller ähnlicher und unähnlichere Punkte ($d_{i,j} > x_{\text{ave}}$) schneller unähnlicher. Es soll nun gelten

$$
\begin{aligned}
& 0,5 = \exp\left(-\frac{1}{2\sigma^2}x_{\text{ave}}^2\right) \\
\Leftrightarrow \quad & \ln(0,5) = -\frac{1}{2\sigma^2}x_{\text{ave}}^2 \\
\Leftrightarrow \quad & 2\sigma^2 = -\frac{1}{\ln(0,5)}x_{\text{ave}}^2 \\
\Leftrightarrow \quad & \sigma^2 = -\frac{1}{2 * \ln(0,5)}x_{\text{ave}}^2 \\
\Leftrightarrow \quad & \sigma = \sqrt{-\frac{1}{2 * \ln(0,5)}x_{\text{ave}}^2}
\end{aligned}
$$

und somit kann der Parameter σ geschätzt werden. Eine andere Methode um σ zu schätzen, wäre den Wendepunkt zu betrachten, denn die Gaußsche Ähnlichkeit besitzt bei $x = \sigma$ einen Wendepunkt. Somit wäre es durchaus sinnvoll $\sigma = x_{\text{ave}}$ zu setzen. Diese zwei Schätzwerte unterscheiden sich jedoch nicht sehr, da

$$
\sqrt{-\frac{1}{2 * \ln(0,5)}x_{\text{ave}}^2} \approx \sqrt{-\frac{1}{2(-0,7)}x_{\text{ave}}^2} = \sqrt{\frac{1}{1,4}x_{\text{ave}}^2} \approx 1,2x_{\text{ave}}
$$

gilt. Wegen der einfacheren Berechnung habe ich mich für die zweite Methode entschieden. Mit dieser Ähnlichkeit kann man etwas wie eine Normalform erreichen, denn die Ähnlichkeiten liegen immer zwischen 1 und 0. Bei der Skalierung der Daten bleiben die Eigenschaften, Unähnlichkeit bei einem größeren Abstand von x_{ave} und Ähnlichkeit bei einem kleineren Abstand als x_{ave}, durch das Neuabschätzen des Parameters σ erhalten.

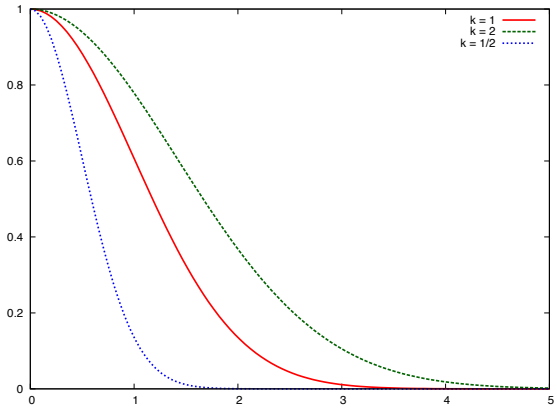

Abbildung 7.3.: $s(x) = \exp(-(\frac{1}{2k^2})x^2)$

7.4.2. Der Datensatz Golub

Der sehr bekannte Datensatz von Golub et al. [3] wurde schon mit verschiedensten Methoden untersucht (s. McLachlan et al. [14]). In Golub et al. [8] geht es um eine Krebsdiagnose und darum, neue Krebsarten zu entdecken. Dazu wurden 38 Patienten Knochenmarksproben entnommen und diese mit Hilfe eines Microarrays analysiert. Als Resultat erhielt man 38 Datenpunkte mit mehr als 7000 Merkmalen. Bei diesen Patienten wusste man über ihre jeweilige Krebsart bescheid, daher enthalten diese Daten insgesamt drei verschiedene Krebsarten. Die zwei Obergruppen hierbei sind die akute lymphatische Leukämie (ALL) mit 27 Datenpunkten und die akute myeloische Leukämie (AML) mit 11 Datenpunkten, wobei sich ALL noch in B–ALL mit 19 Datenpunkten und T–ALL mit 8 Datenpunkten untergliedert. Der *NormalCut* und der *MinMaxCut* lösen diese Aufgabe nicht schlecht, wohingegen der *RatioCut* versagt. Deshalb werden nur die Resultate des *NormalCut* und des *MinMaxCut* dargestellt.
Betrachtet man eine Partition, so kann man durch eine Neuzuordnung eines Punktes, bei k Cluster, $k-1$ neue Partitionen erzeugen. Da es n viele Punkte gibt, kann man so einen gerichteten, gewichteten Graphen $G = (V, E, f)$ erzeugen (diesmal sind negative Kantengewichte erlaubt), bei dem jeder Knoten (Partition) mit $(k-1) * n$ anderen Knoten verbunden ist, und der als Gewichtsfunktion $f(n_1, n_2) := \mathrm{CK}(n_1) - \mathrm{CK}(n_2)$ hat, wobei CK ein Clusterkriterium ist und $n_1, n_2 \in V$ Partitionen darstellen. Man hat somit einen Graphen vor sich, der sowohl positive, als auch negative Kantengewichte beinhaltet. Läuft man an einer Kante (n_1, n_2) mit positiven Kantengewicht entlang, so erreicht man eine Partition n_2 die bezüglich CK einen geringeren Wert hat und somit eine bessere Partition darstellt. Die lokale Suche (Optimierung) benutzt diesen Graphen, um von einem Startpunkt zu einem lokalen Optimum zu gelangen. Wie man dabei die Kanten auswählt, ist vom jeweiligen Algorithmus abhängig. In dieser Arbeit wird der steilste Abstieg verwendet, das heißt, es wird an den Kanten mit den größten Kantengewichten

entlanggelaufen. Eine Alternative dazu wäre zum Beispiel ein randomisierter Ansatz mit mehrmaligem starten.
Die von mir erzielten Resultate sind in den Tabellen 7.1 und 7.2 dargestellt, wobei die Optimierung der lokalen Suche mit dem steilsten Abstieg, und der Partition des spektralen Clusterns als Startwert, entspricht.

Clusterkriterium	von 27 (ALL)	von 11 (AML)	Erfolg
NormalCut	27	4	81,6%
NormalCut mit Optimierung	23	8	81,6%
MinMaxCut	22	8	78,9%
MinMaxCut mit Optimierung	18	9	73,7%

Tabelle 7.1.: Resultate im Überblick für zwei Cluster (Golub)

Clusterkriterium	von 19 (B–ALL)	von 8 (T–ALL)	von 11 (AML)	Erfolg
NormalCut	12	6	8	68,4%
NormalCut mit Optimierung	15	6	11	84,2%
MinMaxCut	12	6	8	68,4%
MinMaxCut mit Optimierung	10	6	10	68,4%

Tabelle 7.2.: Resultate im Überblick für drei Cluster (Golub)

Das natürliche Clusterverhalten und ein spektraler Sprung (Abb. 7.4) weisen auf vier Cluster hin. Auch Golub et al. [8] betrachtet diesen Fall und verweist darauf, dass die Clusteranzahl nicht unbedingt von klinischem Interesse sein muss.

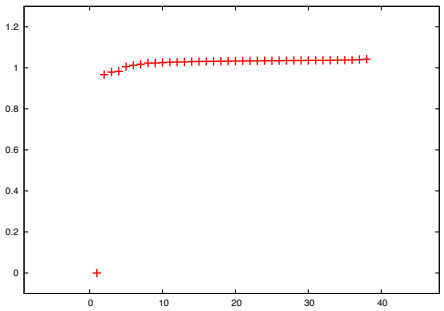

Abbildung 7.4.: Die Eigenwerte bezüglich L_{sym}

7.4.3. Die Datensatzsammlung Stegmaier Leukemia

Als nächste Datensatzsammlung [3] betrachten wir drei Datensätze von Stegmaier et al. [26]. Sie enthalten sowohl Proben von Leukemiepatienten und gesunden Menschen, als auch von Patienten die mit unterschiedlichen Methoden behandelt wurden (Tabelle 7.3), und untersuchen somit unter anderem verschiedene Behandlungsmethoden. Mit Hilfe einer Partition könnte man untersuchen, ob die Behandlungen gezielt das erkrankte Gewebe verändert haben.

AML	primary patient acute myeloid leukemia cells
APL	primary acute promyelocytic leukemia cells
HL–60	Human promyelocytic leukemia cells
undiff	allegemeine HL–60 Zellart
mono	normale menschliche monocytes
poly	normale menschliche neutrophils
ATRA	behandelte HL–60 Zellart
PMA	anders behandelt als *ATRA*

Tabelle 7.3.: Abkürzungen

7.4.3.1. Myeloid APL compound

Dieser Datensatz (Stegmaier et al. [26] [3]) enthält AML (6 Proben), poly (3 Proben) und APL (12 Proben). Somit untersucht er, ob die Krankheiten zum einen untereinander differenzierbar sind (3–Clusterfall), und zum anderen vom gesunden Gewebe zu unterscheiden sind (2–Clusterfall). Mein Ergebnis für zwei Cluster ist in Tabelle 7.4 einsehbar.

Clusterkriterium	von 18 (AML + APL)	von 3 (poly)	Erfolg
NormalCut	18	3	100%
NormalCut mit Optimierung	18	3	100%
MinMaxCut	16	3	90,5%
	von 9 (AML + poly)	von 12 (APL)	
MinMaxCut mit Optimierung	9	10	90,5%

Tabelle 7.4.: Resultate im Überblick für zwei Cluster (Myeloid APL compound)

Der *RatioCut* (mit und ohne Optimierung) isoliert nur einen Datenpunkt, weshalb die Ergebnisse an dieser Stelle nicht aufgelistet werden. Ich möchte hier nochmals darauf

hinweisen, dass nicht versucht wird einen Datenpunkt einer Krankheit zuzuordnen, sondern Krankheiten zu unterteilen. Deshalb werden immer die erfolgreichsten Partitionen aufgelistet (hier einmal AML + APL, und einmal AML + poly). Der *NormalCut* sagt uns also, dass AML und APL unterscheidbar von gesundem Gewebe sind.
Die Ergebnisse für den 3–Clusterfall werden in Tabelle 7.5 dargestellt.

Clusterkriterium	von 12 (APL)	von 6 (AML)	von 3 (poly)	Erfolg
NormalCut	12	6	3	100%
NormalCut mit Optimierung	12	6	3	100%
MinMaxCut	12	6	3	100%
MinMaxCut mit Optimierung	7	2	3	57%

Tabelle 7.5.: Resultate im Überblick für drei Cluster (Myeloid APL compound)

Wie bei dem 2–Clusterfall versagt auch hier der *RatioCut* vollkommen, indem er zwei einzelne Punkte isoliert. Das natürliche Clusterverhalten weist auf ganze sechs Cluster hin. Clustert man in sieben Clustern, so wird ein Punkt isoliert, was auf eine Überanpassung hindeutet. Ein spektraler Sprung weist auf zwei bis drei Cluster hin (Abb. 7.5). Wüsste man nicht, dass es hier um zwei Krankheiten geht, sondern würde man nur die Krankheit Leukemie untersuchen, so würde der *NormalCut* erfolgreich darauf hindeuten, dass diese Krankheit unterteilt werden kann.

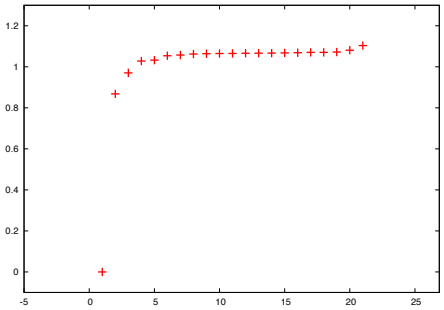

Abbildung 7.5.: Die Eigenwerte bezüglich L_{sym}

7.4.3.2. Myeloid primarycells

Bei diesem Datensatz (Stegmaier et al. [26] [3]) handelt es sich um einen balancierten Datensatz, dass heißt alle Cluster sind gleich groß. Hierbei werden AML, poly und mono mit jeweils drei Datenpunkten betrachtet. Bei drei Clustern erwartet man, dass hier der

MinMaxCut seine Stärke (bevorzugt gleich große Cluster) zeigen kann. Die Ergebnisse des 2–, beziehungsweise des 3–Clusterfalles sind in den Tabellen 7.6 und 7.7 dargestellt.

Clusterkriterium	von 3 (poly)	von 6 (AML + mono)	Erfolg
NormalCut	3	6	100%
NormalCut mit Optimierung	3	6	100%
MinMaxCut	3	6	100%
	von 3 (AML)	von 6 (poly + mono)	Erfolg
MinMaxCut mit Optimierung	3	5	89%

Tabelle 7.6.: Resultate im Überblick für zwei Cluster (Myeloid primarycells)

Clusterkriterium	von 3 (AML)	von 3 (poly)	von 3 (mono)	Erfolg
NormalCut	3	3	3	100%
NormalCut mit Optimierung	3	3	3	100%
MinMaxCut	3	3	3	100%
MinMaxCut mit Optimierung	3	3	3	100%

Tabelle 7.7.: Resultate im Überblick für drei Cluster (Myeloid primarycells)

Auch hier versagt der *RatioCut* und isoliert wieder Datenpunkte. Der *MinMaxCut* stellt bei diesem Datensatz sein theoretisches Verhalten (Tendenz zu gleich großen Clustern) unter Beweis. Überraschend ist, dass die Optimierung bei dem *MinMaxCut* ein schlechteres Resultat bringt als der nicht optimierte Algorithmus.
Ein spektraler Sprung weist auf drei Cluster hin (Abb. 7.6), während das natürliche Clusterverhalten auf sechs Cluster hindeutet.

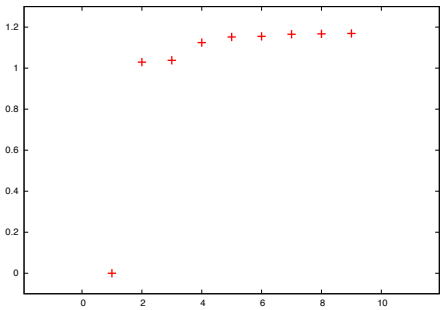

Abbildung 7.6.: Die Eigenwerte bezüglich L_{sym}

7.4.3.3. Myeloid HL–60 undiff PMA ATRA

Bei diesem letzten Datensatz (Stegmaier et al. [26] [3]) handelt es sich um behandelte HL–60 Zellarten, wobei es nur zwei Cluster (9 Proben ATRA und 12 Proben PMA) gibt. Die Ergebnisse sind in Tabelle 7.8 dargestellt.

Clusterkriterium	von 9 (ATRA)	von 12 (PMA)	Erfolg
NormalCut	8	8	76, 1%
NormalCut mit Optimierung	8	8	76, 1%
MinMaxCut	8	8	76, 1%
MinMaxCut mit Optimierung	6	8	66%

Tabelle 7.8.: Resultate im Überblick für zwei Cluster (Myeloid HL–60 undiff PMA ATRA)

Hier weist ein spektraler Sprung auf zwei Cluster hin (Abb. 7.7), was durch das natürliche Clusterverhalten bestätigt wird.

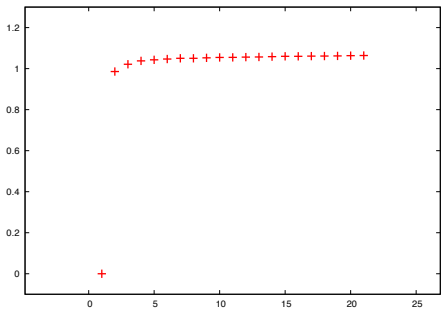

Abbildung 7.7.: Die Eigenwerte bezüglich L_{sym}

7.5. Die alternierende Erweiterung

Wie man sehen konnte clustert der *NormalCut*, mit oder ohne Optimierung, von allen drei Clusterkriterien am erfolgreichsten. Bei dem Datensatz von Myeloid HL–60 undiff PMA ATRA und Golub et al., wurde noch kein perfektes Ergebnis erzielt. Die Frage

ist also, ob es ein Verfahren gibt, welches uns ermöglicht Störfaktoren zu eliminieren. Oft wird in Arbeiten, wie zum Beispiel in Qiu und Plevritis [17] erwähnt, dass man die aussagestärksten 1000 Gene (Merkmale) verwendet um zu clustern. Diese Merkmalsselektion geschieht jedoch oft durch das Wissen über die richtige Partition. So werden diejenigen Merkmale selektiert, mit geringer Standardabweichung und weit auseinander liegenden Zentren der Cluster (s. Golub et al. [8]). Wir wollen auf diese a–priori Information verzichten und versuchen die interessanten Gene zu finden, oder zumindest die störenden Gene zu entfernen. Hierzu habe ich mir ein Verfahren überlegt, welches auf *Bagging* basiert.

7.5.1. Bagging

Um den Begriff *Bagging* zu erklären, wird zunächst der Begriff des *Bootstrapping* betrachtet. Er stammt aus dem Fachbereich Statistik und ist eine Methode des Resampling, wobei die theoretische Verteilung unbekannt ist. Der Begriff *Bagging* ist aus den Worten *Bootstrap* und *Aggregation* zusammengesetzt, und berechnet aus mehreren *Bootstrap* Stichproben eine Verteilung.
Bagging gehört zu den *Ensemble Methoden*, die versuchen die Ergebnisse von vielen **Klassifikationen** zusammenzufassen. Nun kann man zunächst Clustern als Klassifikation ansehen, denn der einzige Unterschied zwischen Klassifikation und Clustern besteht darin, dass die Regeln bei der Klassifikation bekannt sind, jedoch bei dem Clustern zusätzlich bestimmt werden müssen. Kapselt man dieses „Bestimmen“ und nennt es Regel, so hat man eine Klassifikation.
Um verschiedene Modelle einer Klassifikation zu bekommen, werden Stichproben aus den Merkmalen der Microarray–Daten gezogen, und mit diesen gezogenen Merkmalen wird eine Klassifikation erzeugt (also geclustert). Aus den verschiedenen Klassifikationen werden einige **ausgewählt** und zu **einer** Klassifikation aggregiert. Dies ist der Schritt, der es uns ermöglicht, zumindest die störenden Merkmale (Gene) auszusortieren.

7.5.2. Das Bagging basierte Verfahren

Zunächst ist zu sagen, dass der Algorithmus eine Abwandlung des Algorithmus von Dy und Brodley [7] ist. Sie selektieren zufällige Merkmale, clustern diese und bewerten die entstandenen Partitionen durch ein Kriterium. Überschreitet ein Kriteriumswert einen Schwellwert, so werden die Merkmale (Gene) in einer finalen Menge gespeichert.
Der hier vorgestellte Algorithmus unterscheidet sich in der Auswahl der „richtigen“ Partitionen. Bevor eine Auswahl getroffen wird, werden zunächst zahlreiche Partitionen, durch zufällige Merkmalsselektion, Transformation in einen ungerichteten, positiv gewichteten Graphen und anschließendem clustern, erzeugt. Hat man alle Ergebnisse vorliegen, so kann man die Häufigkeit einer generierten Partition zählen und somit die Partitionen

ordnen. Die Partitionen die am häufigsten generiert wurden, sind die mit hoher Wahrscheinlichkeit auf eine „richtige“ Partition. Da die störenden Merkmale (Gene) diejenigen sind, welche **eine** Verteilung, oder mehrere **stark überlappende** Verteilungen haben (s. Abb. 7.8). Dagegen haben die wichtigen Merkmale zumindest **zwei** Verteilungen (eine pro Cluster), welche sich nur **gering oder gar nicht** überlappen (s. Abb. 7.9). Da es bei den störenden Merkmalen nur eine „grobe Verteilung“ gibt (mehrere überlappende Verteilungen agieren ähnlich zu einer Verteilung), gibt es keine Struktur (Partition) die eingehalten wird (fast uniforme Verteilung auf den Partitionen). Selektiert man mehrere störende Merkmale, so wird sich die Struktur (Partition) bei jedem einzelnen Durchlauf ändern. Selektiert man jedoch viele wichtige Merkmale und nur wenige Störende, so wird sich die Struktur (Partition) der wichtigen Merkmale immer wieder zeigen, und somit immer wieder generiert (Dirac ähnliche Verteilung auf den Partitionen). Dies ist nun die Begründung, weshalb man durch die Häufigkeit einer Partition Rückschluss auf deren „Richtigkeit“ erlangen kann. Die einzige Anforderung an die Daten ist, dass die Datenpunkte in den verschiedenen Dimensionen unterschiedliche Anordnungen in $\mathbb{R}$ haben (eine Unabhängigkeit der Verteilungen ist nicht gefordert), da sonst auch die störenden Merkmale immer ähnliche Partitionen erzeugen.

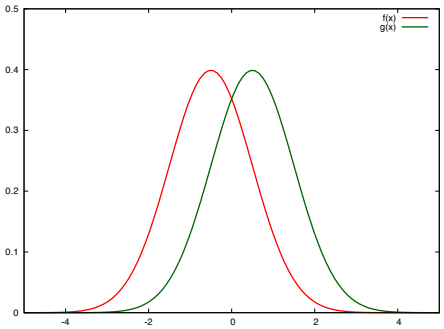

Abbildung 7.8.: Dichtefunktionen zweier stark überlappender Verteilungen

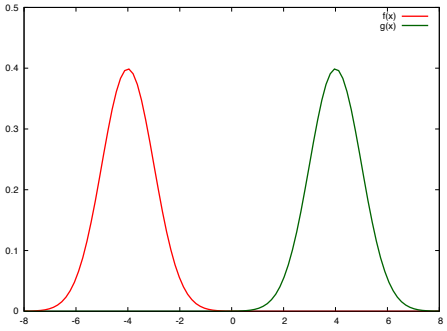

Abbildung 7.9.: Dichtefunktionen zweier kaum überlappender Verteilungen

Eine Möglichkeit um wichtige Partitionen zu selektieren wurde aufgezeigt. Unter den entsprechenden Merkmalen befinden sich natürlich immer noch viele Störende, die jetzt aussortiert werden sollen. Betrachtet man eine Partition die mehrmals generiert wurde, so ist anzunehmen, dass die wichtigen Merkmale öfter als die störenden Merkmale selektiert wurden, da gerade die Wichtigen zu der Partition beigetragen haben. Was also gemacht wird ist, die Merkmale zu selektieren, die am häufigsten bei einer Partition ausgewählt wurden. Dadurch werden ein weiteres mal störende Merkmale aussortiert. Unterscheiden sich zwei Partitionen lediglich um einen Punkt, so ist stark zu vermuten, dass diese zwei Partitionen zusammengehören. Auf diese Art und Weise können transitive Ketten entstehen, die in der Realität jedoch keine große Länge haben, da die Anzahl der wichtigen Partitionen stark begrenzt ist. Hat man eine solche transitive Kette vor sich und will die „richtige“ Partition bestimmen, so selektiert man die wichtigsten Merkmale jeder Partition und clustert ein letztes mal. Die dabei generierte Partition ist ähnlich zu denen, aus welchen sie entstanden ist und kann als die wahrscheinlichste, „richtige“ Partition betrachtet werden. Der gesamte Ablauf des Algorithmus ist in Abbildung 7.10 nochmals dargestellt.

Leider kann man die Einstellungen der Parameter nicht automatisieren. Statt dessen versucht man, diese so zu justieren, dass bei dem Verstellen eines Parameters das Ergebnis trotzdem stabil bleibt. So sind einzustellen:

- Die Anzahl der Wiederholungen zufälliger Clusterungen,
- die Anzahl der Merkmale, die in einem Durchlauf selektiert werden,
- das Auswählen der wichtigsten Partitionen,
- die Anzahl der zu selektierenden, besten Merkmale.

Der kritischste Parameter unter diesen ist die Anzahl der Merkmale, die zufällig selektiert werden. Nach einigen Versuchen an dem Datensatz Golub et al. ist das stabilste Ergebnis bei rund 1000 Merkmalen zu finden. Dies liegt wahrscheinlich daran, dass wie man in Golub et al. [8] nachlesen kann, 1100 Gene aussagekräftig sind. Setzt man diesen Parameter hoch, so sinkt die Anzahl der notwendigen Wiederholungen. Senkt man ihn, so steigt die Anzahl wieder an. Die Anzahl der zu selektierenden besten Merkmale ist intuitiv bei der Hälfte der Anzahl der zufällig gewählten Merkmale anzusiedeln. Zuletzt bleibt das Selektieren der besten Partitionen. Hierbei gibt es zwei verschiedene Vorgehensweisen: Entweder wählt man alle Partitionen, die dank ihrer wiederholten Generierung wichtig sind aus (alle Partitionen die mehr als x mal generiert wurden), oder man fügt ähnliche Partitionen zusammen. Als Beispiel wird der Datensatz Golub et al. [8] mit diesem Verfahren analysiert.

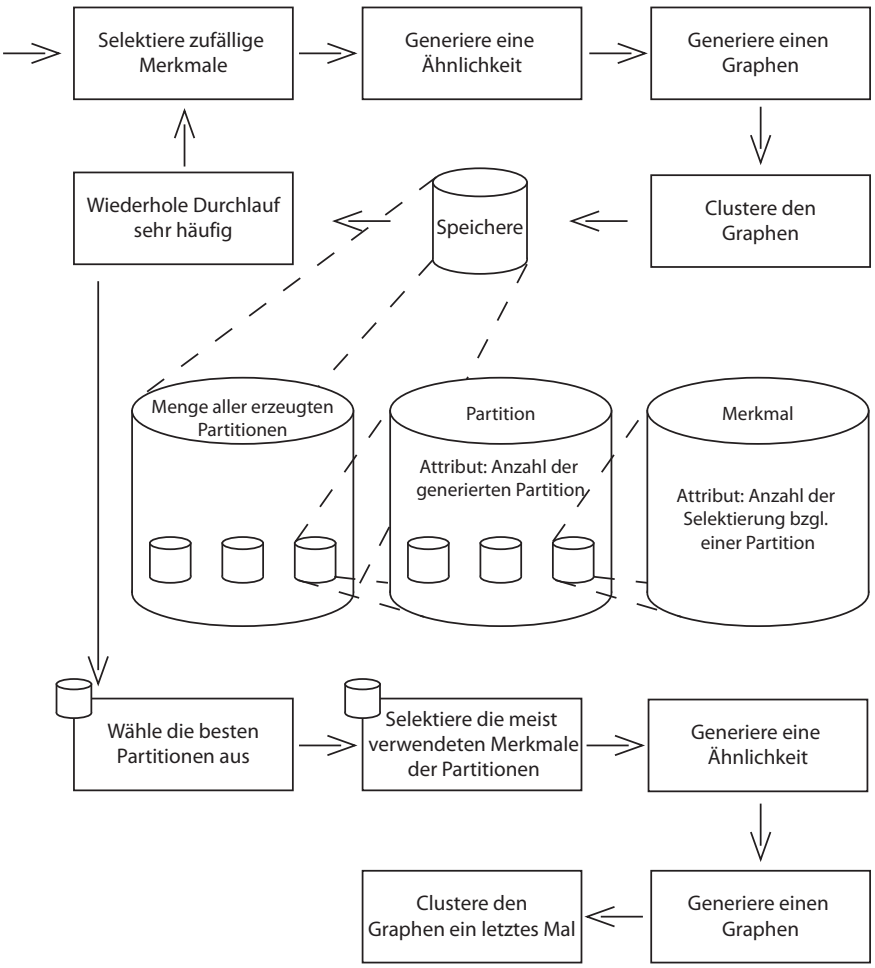

Abbildung 7.10.: Ablaufdiagramm des Bagging basierten Verfahrens.

7.5.2.1. Die Anwendung des Verfahrens auf den Datensatz Golub

Mit Hilfe des oben erwähnten Verfahrens wird versucht, den Datensatz von Golub et al. [8] besser zu clustern. Nach vielen Versuchen haben sich die Parameter 500 Wiederholungen, 1000 zufällige Merkmale und die Selektion der jeweils besten 500 Merkmale als sehr stabil erwiesen. Bei diesen 500 Wiederholungen wurden zehn verschiedene Partitionen (mit Anzahl der Generierungen ≥ 2) generiert.

```
1    Es wird bezüglich NormalCut mit Optimierung geclustert.
2
3    Index 0. Es wurde 30 mal folgende Partition generiert!
4    Cluster 0: 37 36 35 34 33 32 31 30 29 28 27 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
5    Cluster 1: 26 25 24 23 22 21 20 19
6    Index 1. Es wurde 23 mal folgende Partition generiert!
7    Cluster 0: 37 36 35 34 33 32 31 30 29 28 27 25 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
8    Cluster 1: 26 24 23 22 21 20
9    Index 2. Es wurde 13 mal folgende Partition generiert!
10   Cluster 0: 36 35 33 32 31 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
11   Cluster 1: 37 34 30 29
```

```
    Index 3. Es wurde 12 mal folgende Partition generiert!
    Cluster 0: 37 36 35 34 33 32 31 30 29 27 25 19 17 16 15 14 12 11 10 9 7 6 5 4 2 1 0
    Cluster 1: 28 26 24 23 22 21 20 18 13 8 3
    Index 4. Es wurde 8 mal folgende Partition generiert!
    Cluster 0: 37 36 35 34 33 32 30 29 28 27 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
    Cluster 1: 31 26 25 24 23 22 21 20 19
    Index 5. Es wurde 6 mal folgende Partition generiert!
    Cluster 0: 37 36 35 34 33 32 31 30 29 28 27 25 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
    Cluster 1: 26 24 23 22 21 20 19
    Index 6. Es wurde 5 mal folgende Partition generiert!
    Cluster 0: 33 28 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
    Cluster 1: 37 36 35 34 32 31 30 29 27
    Index 7. Es wurde 4 mal folgende Partition generiert!
    Cluster 0: 35 33 28 26 25 24 23 22 21 20 18 17 15 14 13 11 10 8 7 6 5 4 3 2 1 0
    Cluster 1: 37 36 34 32 31 30 29 27 19 16 12 9
    Index 8. Es wurde 3 mal folgende Partition generiert!
    Cluster 0: 37 36 34 33 32 31 30 29 26 23 22 21 19 16 15 13 12 11 10 8 7 6 5 4 2
    Cluster 1: 35 28 27 25 24 20 18 17 14 9 3 1 0
    Index 9. Es wurde 2 mal folgende Partition generiert!
    Cluster 0: 35 28 27 25 20 18 17 14 9 3 1 0
    Cluster 1: 37 36 34 33 32 31 30 29 26 24 23 22 21 19 16 15 13 12 11 10 8 7 6 5 4 2
```

Ausser bei den Partitionen mit Index 4 und 5, welche nur 8 und 6 mal generiert wurden, unterscheiden sich alle Partitionen um mindestens zwei Punkte, weshalb nur die Partition mit Index 0 ausgewählt wird. Tatsächlich entspricht diese Partition einem hundertprozentigem Erfolg. Selektiert man stattdessen die Partitionen mit Index 0, 4, 5 und 1 (transitive Kette), so werden zwei Punkte falsch zugeordnet und man hat somit einen Erfolg von 94,7 %. Es sei hier erwähnt, dass mit diesem Verfahren versucht wird, kleine Cluster zu identifizieren. Geht man von Cluster 1 mit Index 0 aus und ändert man zwei Punkte ab, so verändert man das Cluster um $100\%/8 * 2 = 25\%$. Deshalb begrenzt man die Ähnlichkeit von Partitionen auf einen Punkt. Die Partition mit Index 2 lässt bereits hier erahnen, dass es noch ein weiteres kleines Cluster gibt.
Um bessere Aussagen über die restlichen Cluster treffen zu können, werden die Datenpunkte von Cluster 1 mit Index 0 aus dem Datensatz von Golub et al. [8] entfernt. Wiederholt man obiges Verfahren, so wird das Cluster 1 mit Index 0 nicht mehr „stören“ und die restliche Struktur zum Vorschein kommen.[1]

```
    Es wird bezüglich NormalCut mit Optimierung geclustert.

    Index 0. Es wurde 30 mal folgende Partition generiert!
    Cluster 0: 28 27 25 24 23 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
    Cluster 1: 29 26 22 21
    Index 1. Es wurde 19 mal folgende Partition generiert!
    Cluster 0: 25 20 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
    Cluster 1: 29 28 27 26 24 23 22 21 19
    Index 2. Es wurde 16 mal folgende Partition generiert!
    Cluster 0: 20 18 17 16 15 14 13 12 11 10 8 7 6 5 4 3 2 1 0
    Cluster 1: 29 28 27 26 25 24 23 22 21 19 9
    Index 3. Es wurde 15 mal folgende Partition generiert!
    Cluster 0: 20 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
    Cluster 1: 29 28 27 26 25 24 23 22 21 19
    Index 4. Es wurde 13 mal folgende Partition generiert!
    Cluster 0: 25 20 18 17 16 15 14 13 12 11 10 8 7 6 5 4 3 2 1 0
    Cluster 1: 29 28 27 26 24 23 22 21 19 9
    Index 5. Es wurde 11 mal folgende Partition generiert!
    Cluster 0: 25 23 20 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
    Cluster 1: 29 28 27 26 24 22 21 19
    Index 6. Es wurde 10 mal folgende Partition generiert!
    Cluster 0: 27 25 20 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
    Cluster 1: 29 28 26 24 23 22 21 19
    Index 7. Es wurde 9 mal folgende Partition generiert!
    Cluster 0: 27 25 20 18 17 16 15 14 13 11 10 8 7 6 5 4 3 2 1 0
    Cluster 1: 29 28 26 24 23 22 21 19 12 9
```

[1] Dieses Vorgehen unterstützt nebenbei auch das Schätzen des Parameters für die Gaußsche Glockenkurve, da die Schätzung von zwei Clustern ausgeht.

```
27    Index 8. Es wurde 8 mal folgende Partition generiert!
28    Cluster 0: 27 25 20 18 17 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
29    Cluster 1: 29 28 26 24 23 22 21 19 16
30    Index 9. Es wurde 7 mal folgende Partition generiert!
31    Cluster 0: 25 20 18 17 16 15 14 13 11 10 8 7 6 5 4 3 2 1 0
32    Cluster 1: 29 28 27 26 24 23 22 21 19 12 9
33    Index 10. Es wurde 6 mal folgende Partition generiert!
34    Cluster 0: 27 20 19 18 17 14 9 3 1 0
35    Cluster 1: 29 28 26 25 24 23 22 21 16 15 13 12 11 10 8 7 6 5 4 2
36    Index 11. Es wurde 5 mal folgende Partition generiert!
37    Cluster 0: 27 25 20 19 18 17 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
38    Cluster 1: 29 28 26 24 23 22 21 16
39    Index 12. Es wurde 4 mal folgende Partition generiert!
40    Cluster 0: 27 25 23 20 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
41    Cluster 1: 29 28 26 24 22 21 19
42    Index 13. Es wurde 3 mal folgende Partition generiert!
43    Cluster 0: 29 28 27 26 25 24 23 22 21 19 16 15 13 12 11 10 9 8 7 6 5 4 2 0
44    Cluster 1: 20 18 17 14 3 1
45    Index 14. Es wurde 2 mal folgende Partition generiert!
46    Cluster 0: 29 28 27 26 25 24 23 22 21 19 16 5
47    Cluster 1: 20 18 17 15 14 13 12 11 10 9 8 7 6 4 3 2 1 0
```

Diesmal ist es etwas kniffliger: Betrachtet man die Partitionen mit Index 1–7, so kann man einen Baum ihrer Ähnlichkeit erstellen (s. Abb. 7.11).
Die Partitionen ab Index 7 sind zu verschieden, oder haben durch ihre relative Häufigkeit in ihrer Generierung keine große Bedeutung. Werden die relativen Häufigkeiten der Generierungen von 1–7 summiert, so kommt man auf 93 Wiederholungen. Im Vergleich dazu hat die Partition mit Index 0 gerade einmal 30 Wiederholungen, was ein Indiz dafür ist, dass die wirkliche Partition in 1–7 verborgen liegt. Man fasst also die erfolgreichsten Merkmale (wurden am häufigsten verwendet) von 1–7 zusammen (3924 Merkmale) und clustert ein letztes Mal. Als Resultat erhält man eine Partition, die lediglich einen Punkt falsch zuordnet, was einem Erfolg von 97,4 % entspricht.
Zusammenfassend bleibt zu sagen, dass durch die Anwendung dieses Verfahrens der Erfolg erheblich verbessert werden konnte. Die Tatsache dass es bei diesem Beispiel gut funktioniert hat, heißt natürlich nicht, dass dies auch für alle anderen Daten gilt. Einen Einblick in die tiefere Struktur eines Datensatzes erhält man aber allemal.

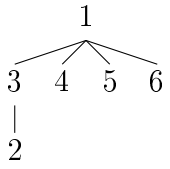

Abbildung 7.11.: Die Partitionen der Väter zu Ihren Söhnen unterscheiden sich um einen Punkt.

7.5.2.2. Die Anwendung auf Myeloid HL–60 undiff PMA ATRA

Das Verfahren verfehlt leider seine Wirkung bei diesem Datensatz, was man schon daran erkennen kann, dass bei den Wiederholungen immer ähnliche Partitionen generiert

werden, und diese Partitionen nahe bei der Standardpartition liegen (Partition ohne alternierende Erweiterung). Das kann bedeuten, dass man immer wieder die störenden Merkmale selektiert und somit zu viele Stichproben bei den Merkmalen auswählt. Setzt man jedoch die Stichprobenanzahl auf nur 100 Merkmale und selektiert jeweils nur die besten 50, so ändert sich das Ergebnis nur marginal.
Als Konsequenz bleibt anzumerken, dass scheinbar die innere Struktur der Daten nicht die gewünschte Struktur wiedergibt, zumindest nicht unter den gegebenen Merkmalstransformationen.

7.6. Die Überanpassung

Bei dem spektralen Clustern und bei den Modellannahmen (Merkmalstransformation), wurde nie ein Parameter von Hand eingestellt. Dies spricht dafür, dass ein Modell geschaffen wurde, welches auch für andere Daten gut funktioniert, da die benötigten Einstellungen automatisch rekalibriert werden. Erst bei der Bagging basierten Erweiterung, findet man Parameter die von Hand eingestellt werden müssen. Da aber dieses Verfahren gerade auf der Stabilität der Partitionen beruht, findet selbst hier keine Überanpassung statt. Im Gegenteil, das Verfahren versichert uns, dass eine Partition nicht durch eine zufällige Überanpassung entstanden ist.
Allgemein wurden die Modellannahmen auf unabhängige Daten mit Erfolg angewendet, was für eine gute Generalisierung spricht.

7.7. Der Vergleich des Ergebnisses mit anderen Arbeiten

Ein Vergleich mit anderen Arbeiten fällt sehr schwer, da Qiu und Plevritis [17] überwachte Klassifikation und Clusterung simultan betrachten, weshalb sie Wissen über die Partition indirekt mit einfließen lassen (sie clustern mit den besten 1000 Merkmalen, was Vorabwissen über die Partition bedingt), oder andere Clusterkriterien verwendet werden. Um jedoch etwas über die Güte auszusagen, wird Tritchler et al. [28], Golub et al. [8] und Qiu und Plevritis [17] mit meinen Ergebnissen verglichen. Tritchler benützt eine vorverarbeitete Teilmenge aller Merkmale (3571 Gene), wobei leider nicht beschrieben wird wie diese Teilmenge zustande kommt (wurde Wissen über korrekte Partition verwendet?) und betrachtet nur den 3–Clusterfall mit einem eigenen Clusterkriterium. Golub et al. [8] benutzt eine *SOM* (self–organized–map) um die Daten zu clustern, Qiu und Plevritis [17] eine andere Ähnlichkeitsabbildung. Bei Qiu und Plevritis [17] wird leider nur der 3–Clusterfall betrachtet, der AML unterteilt, jedoch nicht ALL. Um einen

Vergleich anstellen zu können, werden zwei Cluster zusammengefasst. Die Vergleiche für zwei Cluster sind in Tabelle 7.9 einzusehen.

Art	von 27 (ALL)	von 11 (AML)	Erfolg
SOM (s. Golub et al. [8])	24	8	89,5%
SPACC (s. Qiu und Plevritis [17])	25	11	94,7%
NormalCut	27	4	81,6%
NormalCut mit Optimierung	23	8	81,6%
	von 30 (B–ALL + AML)	von 8 (T–ALL)	Erfolg
NormalCut mit Optimierung und Bagging	30	8	100%

Tabelle 7.9.: Vergleich der Resultate vom Datensatz Golub für zwei Cluster

Bei Golub et al. [8] werden nur vier Cluster gebildet, weshalb hier wieder zwei von ihnen zusammengefasst werden. Die Vergleiche kann man in Tabelle 7.10 betrachten.

Art	von 19 (B–ALL)	von 8 (T–ALL)	von 11 (AML)	Erfolg
s. Tritchler et al. [28]	17	8	11	94,7%
SOM (s. Golub et al. [8])	18	8	10	94,7%
NormalCut	12	6	8	68,4%
NormalCut mit Optimierung	15	6	11	84,2%
NormalCut mit Optimierung und Bagging	19	8	10	97,4%

Tabelle 7.10.: Vergleich der Resultate vom Datensatz Golub für drei Cluster

7.8. Die Zusammenfassung

Zusammenfassend kann man sagen, dass spektrales Clustern eine gute Methode ist, um Wissen aus Microarray–Daten zu gewinnen. In diesem Kapitel wurde erarbeitet, dass die Gaußsche Glockenkurve eine sehr gute Methode ist, um eine Ähnlichkeit zu gewinnen, ohne einen Parameter von Hand einstellen zu müssen. Des weiteren wurde festgestellt, dass man den *NormalCut* mit Optimierung verwenden sollte, um bessere Ergebnisse zu erzielen. Störende Merkmale kann man oft filtern, was jedoch etwas Fingerspitzengefühl erfordert, um ein optimales Ergebnis erzielen zu können.

Kapitel 8.

Die Schlussbetrachtung

Diese Diplomarbeit hat sich eingehend mit der Theorie der spektralen Datenanalyse beschäftigt. Dabei wurde gezeigt, was Cluster sind, wozu diese gut sind und wie man diese bildet. Die Grundlage einer guten Analyse ist eine passende Merkmalstransformation, denn nur durch diese kann man die Semantik eines Clusters modellieren. Einem Computer beizubringen, was Semantik ist, wird wohl nie möglich sein. Dadurch wird es wahrscheinlich keine umfassende Arbeit geben, der man die Merkmale übergeben kann, und die uns die passenden Antworten liefert.
Um die Güte einer Partition zu bewerten wurde definiert, was Clusterkriterien sind und verschiedene Clusterkriterien eingeführt, die später approximiert wurden. Die Notwendigkeit der Approximation folgt aus der Anzahl der möglichen Cluster (bei zwei Cluster $2^{n-1}-1$) und der Tatsache, dass ein greedy Algorithmus nicht zwangsweise zu dem **globalen** Optimum führt.
Um eine Annäherung an das **globale** Optimum zu erreichen, wurde der Zusammenhang zwischen den Clusterkriterien und den Eigenvektoren erläutert. Dieser Zusammenhang wird über einen kleinen Umweg über Dirac Vektoren und den speziellen Rayleigh–Quotienten hergestellt. Dabei wurde für den $NormalCut$ der allgemeine Rayleigh–Quotient in den speziellen Rayleigh–Quotienten überführt, wodurch klar wurde, wieso man L_{sym} verwenden muss. Nach der Definition des $MinMaxCut$ und einigen weiteren Umformungen, konnte eine Entwicklung der Clusterkriterien abgelesen werden. Der $MinMaxCut$ wurde inspiriert durch den $NormalCut$, und dieser wiederum durch den $RatioCut$. Damit soll jedoch nicht unterstellt werden, dass der $MinMaxCut$ besser als der $NormalCut$ ist. Meine erste Wahl bei den Clusterkriterien ist der $NormalCut$, da er sich den Daten anpasst und eine Verbindung zu Markoff–Ketten besteht, welche ein theoretisches Fundament darstellt. Dass die Approximation durch Eigenvektoren eine tatsächliche Annäherung ist, wird durch die Perturbationstheorie garantiert.
Die Anzahlbestimmung der Cluster die ein Datensatz enthält, ist immer ein kritischer Prozess. Festzuhalten ist, dass zu kleine Cluster nicht besonders aussagekräftig sind, und dass das natürliche Clusterverhalten die Struktur der Daten beachtet.
Bei unserem Anwendungsgebiet hat uns die Gaußsche Glockenkurve gute Resultate geliefert. Durch eine randomisierte Merkmalsauswahl haben wir mehr über die Struktur

erfahren, was es ermöglichte, störende Merkmale auszusortieren. Auch wenn diese Methode mehr Arbeit erfordert und viele unbekannte Parameter enthält, hat diese Vorgehensweise großes Potential.
Am Anfang der Diplomarbeit wurde bereits über die Semantik der Daten gesprochen. Die Merkmalsauswahl versucht eine beliebige Semantik zu finden und zu bewerten (durch die wiederholte Generierung der Partition). Durch die Ähnlichkeit von Partitionen kann man semantisch wertvolle Partitionen selektieren und diese zusammenführen. Dies ist ein Schritt in Richtung der automatischen Merkmalstransformation. Auch die Anzahlbestimmung der Cluster ist mit der alternierenden Erweiterung möglich.

Literaturverzeichnis

[1] *http://en.wikipedia.org/wiki/Rayleigh_ quotient.*

[2] *http://www.accessexcellence.org/RC/VL/GG/rna2.php.*

[3] *http://www.broadinstitute.org/cancer/software/genepattern/datasets/.*

[4] Fan R. K. Chung. *Spectral Graph Theory.* AMS und CBMS, 1997.

[5] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the first IEEE International Conference on Data Mining (ICDM)*, pages 107–114, 2000.

[6] Chris Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A minmaxcut spectral method for data clustering and graph partitioning. Technical Report 54111, Lawrence Berkeley National Laboratory University of California, Berkeley, CA 94720 Department of Computer Science and Engineering Pennsylvania State University, University Park, PA 16802 Department of Mathematics University of California, Berkeley, CA 94720, 1. Dezember 2003.

[7] Jennifer G. Dy and Carla E. Brodley. Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5:845Ð889, 2004.

[8] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[9] M. Gu, H. Zha, C. Ding, X. He, and H. Simon. Spectral relaxtion models and structure analysis for k-way graph clustering and bi-clustering. Technical report, Penn State Univ Tech Report CSE-01-007, 2001.

[10] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. In *Computer-Aided Design*, volume 11(9), pages 1074–1085, 1992.

[11] H. Heuser. *Lehrbuch der Analysis Teil 1.* B.G. Teubner Stuttgart, 1980.

[12] R.C. Jancey. Multidimensional group analysis. *Australian J. Botany*, 14:127–130, 1966.

[13] U. von Luxburg. A tutorial on spectral clustering. Technical Report 149, Max Planck Institute for Biological Cybernetics, August 2006.

[14] Geoffrey J. McLachlan, Kim-Anh Do, and Christophe Ambroise. *Analyzing Microarray Gene Expression Data*. John Wiley & Sons, Hoboken, N.J., 2004.

[15] Miller McPherson, Lynn Smith-Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[16] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[17] Peng Qiu and Sylvia K. Plevritis. Simultaneouse class discovery and classification of microarray data using spectral analysis. *Journal of Computational Biology*, 16:935–944, 2009.

[18] Kannan Ravi, Vempala Santosh, and Vetta Adrian. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004.

[19] Gunter Ritter. *Grundlagen der Stochastik*. Universität Passau, 2005.

[20] Gunter Ritter. *Markoff-Ketten*. Universität Passau, 2007.

[21] Gunter Ritter. *Datenanalyse*. Universität Passau, 2008.

[22] Susanna Röblitz and Marcus Weber. Fuzzy spectral clustering by pcca+. Technical Report 73-78, WIAS, 2009.

[23] Uwe Schöning. *Theoretische Informatik – kurzgefaßt*. Spektrum Akademischer Verlag, 1997.

[24] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22(8), pages 888–905, 2000.

[25] Bernhard Sick. *Soft-Computing*. Universität Passau, 2005/2006. Vorlesungsskript.

[26] Kimberly Stegmaier, Kenneth N Ross, Sierra A Colavito, Shawn O'Malley, Brent R Stockwell, and Todd R Golub. Gene expression–based high–throughput screening(ge–hts) and application to leukemia differentiation. *Nature Genetics*, 36:257–263, 2004.

[27] Aida Toma. Minimum Hellinger distance estimators for some multivarita models: influence functions and breakdown point results. *C. R. Acad. Sci.*, 345:353–358, 2007.

[28] David Tritchler, Shafagh Fallah, and Joseph Beyene. A spectral clustering method for microarray data. *Computitional Statistics & Data Analysis*, 49:63–76, 2005.

Anhang A.

Ein kleines Gedankenexperiment

Bei meinen Überlegungen zum $NormalCut$, zu seiner Verbindung zu $Markoff-Ketten$ und zu einer Anschauung bezüglich dieser Verbindung, ist mir folgendes eingefallen: In den letzten Jahren haben so genannte *communities* (wie zum Beispiel Facebook) einen starken Boom erlebt. Dort gibt es meistens eine Funktion, die das Schließen von Freundschaften ermöglicht. Dadurch könnte man nun einen ungerichteten, positiv gewichteten Graphen entwickeln, bei dem jeder Knoten einen Benutzer darstellt und zwei Benutzer immer dann miteinander verbunden sind, wenn sie Freunde sind (Kantengewichte entsprechen 1). Diesen Graphen könnte man in eine Irrfahrt transformieren und anschließend mit $NormalCut$ clustern, wobei die Cluster weniger interessant sind als ihre Grenzen. Wenn man die Benutzer ihrer geographischen Position zuordnet, könnte man die Grenzen des Graphen als Grenzen innerhalb eines Landes, oder eines Kontinents betrachten. Würde man diese Anschauung nur für Deutschland heranziehen, so wäre es von sozialem und politischem Interesse, ob sich die ehemalige Grenze zur DDR verschoben hat, bestehen geblieben ist, oder nicht mehr erkennbar ist.
Bei dem zugrunde liegenden Graphen handelt es sich um ein *soziales Netzwerk* und beruht auf der Homophily–Theorie („gleich und gleich gesellt sich gerne“, s. Miller et al. [15]). Des weiteren wird angenommen, dass Freunde, geographisch betrachtet, nahe beieinander leben und diese nicht durch eine Grenze abgehalten werden, sich gegenseitig zu besuchen. Durch das Internet verwischt diese Annahme natürlich, wird aber nicht ganz durchbrochen. Die Qualität der Partition hängt also von den Daten ab, und somit von den ungeschriebenen Regeln einer *community*, wann Freundschaften geschlossen werden. Analog dazu könnte man das Straßennetz als ungerichteten, positiv gewichteten Graphen betrachten. Die Kantengewichte würden dann dem Verkehrsaufkommen entsprechen, und die Resultate sollten ähnlich zu interpretieren sein.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Diplomarbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet habe, sowie dass diese Diplomarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt wurde.

Ort, Datum Franz-Georg Ulmer