

Das Hierarchische Radius-basierte Competitive Learning (HRCL) im Vergleich mit statistischen und neuronalen Clusteranalyseverfahren

Udo Heuser

Prof. Dr. W. Rosenstiel

WSI 99-08

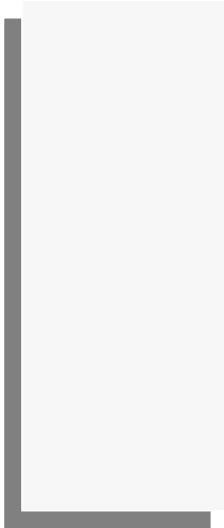
12. Mai 1999

Wilhelm-Schickard-Institut
Universität Tübingen
D-72076 Tübingen, Germany

E-Mail: heuser@informatik.uni-tuebingen.de

© WSI 1999
ISSN 0946-3852

1. Einführung und Motivation	3
2. HRCL	4
2.1. Top-Down HRCL	9
2.2. Bottom-Up HRCL	13
3. Vergleichende Ergebnisse	14
3.1. Eingabedaten	16
3.2. SMART	17
3.3. SOM	18
3.4. Top-Down HRCL	23
3.5. Bottom-Up HRCL	27
4. Diskussion der Ergebnisse	32
5. Zusammenfassung	37
6. Ausblick	38
Anhang A	43
A.1. Notation	43
Anhang B	45
B.1. Ergebnisse SMART	45
B.2. Ergebnisse SOM	46
B.3. Ergebnisse Top-Down HRCL	48
B.4. Ergebnisse Bottom-Up HRCL	49



ANTIPHOLIS. Wer mich meinem eignen Vergnügen überläßt, überläßt mich einem Ding, daß ich nirgends finden kan. Ich bin in der Welt wie ein Tropfen Wassers, der im Ocean einen andern Tropfen suchen will, und indem er hineinfällt sich selbst verliert, ohne den andern zu finden. (...)

(William Shakespeare: Die Irrungen - I. Aufzug, 3. Szene)

1. Einführung und Motivation

Im folgenden soll das Hierarchische Radius-basierte Competitive Learning (HRCL) vorgestellt und mit weiteren (statistischen sowie neuronalen) Clusteranalyse-Verfahren verglichen werden. Clusteranalyse-Verfahren sollen Orte hoher Wahrscheinlichkeitsdichten beliebiger multi-dimensionaler Eingabevektorräume detektieren. Diese Orte hoher Wahrscheinlichkeitsdichten, auch Modi genannt, werden statistisch hinreichend durch die Statistiken erster Ordnung, also Modus-Zentrum (Erwartungswert, Cluster-Zentroid oder Mittelwert) und Kovarianz beschrieben. In vielen echten Anwendungen ist dem Clusteranalyse-Verfahren weder die Anzahl oder Mittelwerte, noch die Verteilung oder Kovarianzen der zugrundeliegenden Modi bekannt. Mehr noch, in der Regel kann das Clusterverfahren noch nicht einmal davon ausgehen, daß die Eingabemodi normal-verteilt vorliegen.

Dieses Unwissen über die Daten führt dazu, daß die eingesetzten (statistischen und neuronalen) Verfahren die beiden Statistiken erster Ordnung selbständig aus der Eingabe gewinnen müssen. Dies bedeutet zugleich für neuronale Netze, daß sie unüberwacht lernen bzw. sich selbst organisieren müssen, denn es gibt keine a-priori bekannten Klassenprototypen, die einem neuronalen Netz als Teaching-Input dienen könnten. Für den realen Fall, in dem angenommen wird, daß nicht notwendigerweise normal-verteilte Eingabemodi existieren, beschränken sich Clusteranalyse-Verfahren auf eine gute Repräsentation der Clusterverteilungen, und plazieren Clusterprototypen mehr oder weniger gleichverteilt an solch „dichte“ Orte. Die zweite Statistik erster Ordnung, die Clustermittelwerte, müssen bei dieser Vorgehensweise immer erst noch aus den „Clusterergebnissen“ herausgelesen werden. Im Ansatz der Selbst-Organisierenden Karte (SOM), als hautsächlicher Repräsentant eines Soft Competitive Learnings mit vorher festgelegter Dimensionalität der Ausgabeschicht, werden Modimittelwerte aus der zweidimensionalen Repräsentation der Nachbarschaftsbeziehungen „herausgelesen“, oder aber es werden Werkzeuge dem eigentlichen Clusteralgorithmus nachschaltet, um Mittelwerte zu errechnen.

Das Hierarchische Radius-basierte Competitive Learning möchte nun sowohl die Modusverteilungen als auch deren Mittelwerte sinnvoll repräsentieren können. Es geht dabei vereinfachend von normalverteilten Eingabemodi aus, die zudem dieselben Kovarianzen besitzen. Diese starke Vereinfachung soll in weiteren hierarchischen Schritten so weit verfeinert werden, daß die nur grob detektierten „primären Cluster“ zu ihren Details, d. h. Subclustern oder Modiverteilungen aufgelöst werden können.

Dabei soll das Verfahren vollkommen automatisiert ablaufen. D. h. es sollen, wegen der Unkenntnis über Anzahl und Verteilung der Eingabemodi, zum einen die Anzahl der initialen Neuronen selbständig bestimmt werden, zum anderen sollen zur Vektorquantisierung überflüssige Neuronen automatisch verworfen werden. Die entstandene Hierarchie soll sowohl die Modizentren, eventuell existierende Submodizentren, als auch deren zugrundeliegenden n -dimensionalen Verteilungen widerspiegeln können.

Im weiteren soll ausführlich das von dem Autor entwickelte Hierarchische Radius-basierte Competitive Learning (HRCL) vorgestellt werden. Im Anschluß daran soll dessen Ergebnisse beispielhaft mit denen eines herausragenden Vetreters sowohl der statistischen Clusteranalyse-Verfahren als auch der unüberwachten neuronalen Netze verglichen werden. Der vorliegende Bericht schließt mit einer Zusammenfassung und einem Ausblick, der vor allem auf die Verwendung des HRCL zur hierarchischen Clusterung multi-dimensionaler HTML-Dokumentensammlungen eingehen soll.

2. HRCL

Gegeben sei eine diskrete und endliche Eingabe-Datenmenge $\mathbf{D} = \{\xi_1, \dots, \xi_M\}$, $\xi_i \in \mathbf{R}^n$, die einer kontinuierlichen multivariaten Signalverteilung $p(\xi)$, erzeugt durch eine kontinuierliche Wahrscheinlichkeitsdichte-Funktion $p(\xi)$, $\xi \in \mathbf{R}^n$, entspricht. Die Signalverteilung $p(\xi)$ sei multi-modal, d. h. bestehe aus mehreren räumlich voneinander getrennten (globulären) Clustern (auch „(globuläre) Modi“ genannt). Globuläre Cluster bestehen aus Subclustern, welche wiederum Subcluster enthalten mögen, u.s.w. Ziel des *Hierarchischen Radius-basierten Competitive Learnings* (HRCL) sei es, sowohl die globulären Cluster als auch deren (Sub-)Subcluster zu detektieren. Hierbei genüge es, im Sinne einer echten Vektorquantisierung, einen einzigen Prototyp (auch „Cluster-Zentroid“ genannt) für jedes existierende (Sub-)Cluster zu erzeugen. In der Terminologie der künstlichen neuronalen Netze bedeutet dies, zum einen so viele Neuronen zu generieren wie (Sub-)Cluster vorhanden sind und zum anderen deren Gewichte so zu adaptieren, daß sie mit den (Sub-)Cluster-Zentroidvektoren zusammenfallen. Im Fall, daß weder Erwartungswerte noch „Form“ oder Verteilung der Cluster bekannt sind (d. h. beim Fehlen eines jeglichen Teaching-Inputs zu einem überwachten Bayeschen- oder Maximum-Likelihood-Estimator; s. [DuHa73], S. 44 ff. bzw. beim Fehler von Teaching-Inputs zu überwachten lernenden neuronalen Netzen), muß das Lernen unüberwacht sein, und das neuronale Netz muß sich selbst organisieren.

Für die oben bezeichnete Clusteranalyse bieten sich zwei Lösungsansätze an: Im ersten Fall versucht das neuronale Netz in einem ersten Schritt die Neuronen zu den Cluster-Zentroiden hin zu trainieren. Falls detektierte Cluster Subcluster enthalten, wird in einem zweiten Schritt versucht diese zu erkennen, indem ein neues neuronales Netz generiert wird, das so viele Neuronen enthalten soll, wie Subcluster vorhanden sind und die so trainiert werden, daß ihre Gewichte mit den Cluster-Zentroiden der Subcluster zusammenfallen. Dies wird für alle (Sub-)Cluster fortgesetzt, bis eine weitere Verfeinerung wegen

fehlender Subcluster nicht mehr möglich ist. Bei Beenden des Verfahrens haben die neuronalen Netze einen Hierarchiebaum erzeugt, an deren Wurzel alle solche Neuronen mit ihren adaptierten Gewichten eingetragen sind, die zur Detektion der „primären“ Cluster beigetragen haben. Jedes der Wurzelneurone bildet so viele Äste, wie Subcluster detektiert wurden, jedes Subcluster besitzt so viele Äste wie Subsubcluster erkannt wurden, u.w.w. Die Hierarchisierung des ersten Ansatzes erfolgt also „Top-Down“, von Clustern zu deren Subclustern und Subsubclustern u.s.w.

In einem zweiten Lösungsansatz, dem „Bottom-Up“, wird versucht kleinstmögliche Wahrscheinlichkeitsdichten im Eingaberaum zu detektieren. Jedes adaptierte Neuron bzw. Neuronengewicht der ersten Bottom-Up-Hierarchiestufe falle mit vorhandenen Cluster-Zentroiden bzw. deren Vektoren zusammen oder versuche die Verteilung der Cluster zu repräsentieren. In der nächsten Hierarchiestufe werden solche Neuronen der ersten Hierarchiestufe zusammengefaßt, die Subcluster-Zentroiden eines ihnen allen gemeinsamen Clusters bezeichnen. Das ihnen allen gemeinsame Cluster wird auch „Supercluster“ genannt. Diese verallgemeinernde Vorgehensweise erfolgt für alle Neuronen der ersten und allen weiteren Hierarchiestufen, falls sie die oben genannte Bedingung erfüllen und bis keine weitere Abstraktion mehr möglich ist.

Die unterschiedlichen Vorgehensweisen der beiden Varianten sollen im folgenden genauer erläutert werden: Sowohl das Top-Down als auch das Bottom-Up HRCL basieren grundsätzlich auf dem „Neuronalen Gas“ (NG) von T. M. Martinetz und K. J. Schulten (s. [HeRo98], S. 41 ff.). Wie dieses ist das HRCL ein Competitive Learning-Verfahren („Wettbewerbslernen“) ohne existierende Netzwerkdimensionalität. Nach jeder zufällig ausgewählten Eingabe ξ , $\xi \in \mathbf{R}^n$ aus der Eingabe-Datenmenge \mathbf{D} werden die vorhandenen HRCL-Neuronen c_i aus der Neuronenmenge $\mathbf{A} = \{c_1, c_2, \dots, c_N\}$, abhängig von den Distanzen ihrer Referenzvektoren $w_{c_i} \in \mathbf{R}^n$ zur aktuellen Eingabe ξ , sortiert.

Das HRCL-Verfahren unterscheidet sich maßgeblich vom Neuronalen Gas durch die Einführung eines Radius' r bzw. eines Hyperkubus' oder hyperkubischen Umgebung $U_c(r) := (2r)^n$ um jedes Neuron c . Diesem Ansatz liegt zugrunde, daß jeder (normal-verteilte) Modus im multivariaten Wahrscheinlichkeitsdichterraum durch seine Statistiken erster Ordnung, also Modus-Zentrum (Erwartungswert $E(\xi)$, Cluster-Zentroid oder Mittelwert μ) und Kovarianz $\Sigma = E((\xi - \mu)(\xi - \mu)^t)$ hinreichend beschrieben wird (s. [DuHa73], S. 22 ff.). Die Kovarianz Σ gibt dabei die Orte des Eingabevektorraums an, an denen die Eingaben ξ gleiche Varianzen bezüglich Modusmittelpunktes μ haben (s. a. Abb. 2.1.). Die durch den Radius r festgelegte Umgebung $U_c(r)$ um jedes Neuron c kann dabei als (grobe) Approximation der Kovarianz Σ angesehen werden, zugleich approximiert das adaptierte Neuronengewicht das Cluster-Zentroid μ . Der vom Benutzer an das System einzugebende Radius r , und damit das Volumen der Umgebung $U_c(r)$ um jedes Neuron c , bleibt dabei im HRCL, wie im Parzen-Window Ansatz, konstant (s. [HeRo98], S. 21 ff. und [DuHa73], S. 22 ff. und S. 88 ff.).

Die Idee des „Rival Penalized Competitive Learnings“ (RPCL; [HeRo98], S. 39 ff.), nicht nur das gewinnende Neuron, sondern auch das zur Eingabe ξ zweitnächste Neuron zu adaptieren, wird im HRCL erweitert: Im HRCL wird nicht nur der Gewinner s_w (mit zugehörigem Gewicht bzw. Neuronenposition w_w) zur Eingabe hin adaptiert, sondern auch alle anderen Neuronen, abhängig von ihren Distanzen zu ξ , entweder zur Eingabe hin adaptiert oder von ihr entfernt. Eine Adaption der HRCL-Neuronen ist dabei immer auch abhängig von der Anzahl der Eingabevektoren, die innerhalb der Umgebung $U_c(r)$ des zu adaptierenden Neurons c zu liegen kommen: Hat die Anzahl der durch die Umgebung $U_{s_w}(r)$ des gewinnenden HRCL-Neurons s_w „eingefangenen“ Eingabevektoren eine Gleichverteilungsschwelle $\Theta(r)$ überschritten, so wird s_w fixiert und nur dann zur aktuellen Eingabe ξ hin adaptiert, falls s_w in der neuen Position $w_{s_w}(t+1)$ mehr Eingabevektoren innerhalb seiner Umgebung sammeln kann, als in der alten Position $w_{s_w}(t)$. Ist dies nicht der Fall, so wird Gewinner s_w nicht adaptiert und behält seine alte Position. Alle weiteren HRCL-Neuronen c_i , $c_i \neq s_w$ werden ebenfalls zur Eingabe hin adaptiert, falls sie mit dem bereits adaptierten Gewinnerneuron nicht in Kon-

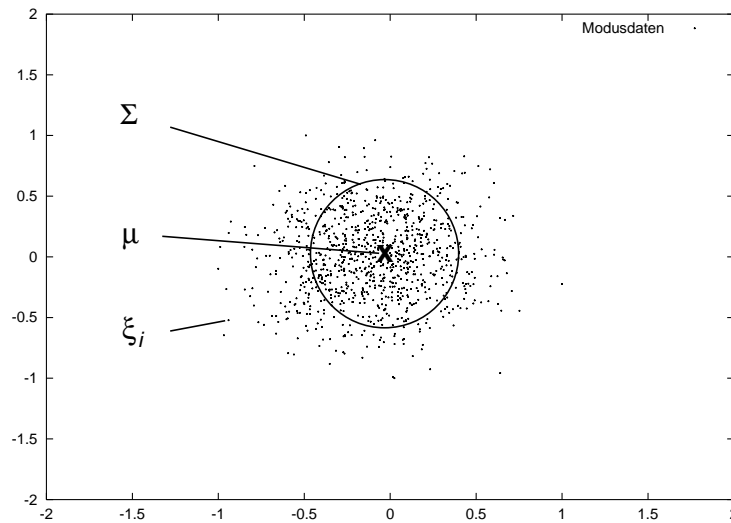


Abb. 2.1.: Charakterisierung eines normal-verteilten bivariaten Modus' durch seine Statistiken 1. Ordnung Mittelwert μ und Kovarianz Σ .

flikt geraten, d. h. falls sich die beiden Umgebungen $U_{s_w(t+1)}(r)$ und $U_{c_i(t+1)}(r)$, ($c_i \neq s_w$) nicht über ein erlaubtes Maß hinweg überlappen. Das erlaubte Maß, der sogenannte „Überlappungskoeffizient“, kann durch den Benutzer definiert werden und ist vom System mit 0 % vorgegeben. Kommt es bei der Adaption zu einem Neuronenkonflikt zwischen Gewinner s_w und einem weiteren HRCL-Neuron c_i , $c_i \neq s_w$, so wird c_i von der aktuellen Eingabe ξ weg bewegt, falls c_i in der alten Position $w_{c_i}(t)$ nicht fixiert ist. War c_i in der alten Position bereits fixiert, d. h. „sammelte“ innerhalb seiner Umgebung $U_{c_i(t)}(r)$ mehr Eingaben als durch die Gleichverteilungsschwelle $\Theta(r)$ festgelegt, so behält c_i seine alte Position und wird nicht von der Eingabe entfernt. Durch die Abhängigkeit der Neuronenadaption von der Anzahl der innerhalb des Neuronenradius' r „eingesammelten“ Eingaben (und in Abhängigkeit von der Gleichverteilungsschwelle $\Theta(r)$) soll erreicht werden, daß Neuronen wirklich nur zu den potentiellen Modi hin bewegt werden, um diese zu erkennen.

Ein weiteres Merkmal des HRCL-Verfahrens ist es, daß sowohl während als auch nach dem Lernen überschüssige HRCL-Neuronen verworfen werden können („Neuronen-Pruning“; s. a. [Reed93] und [Zell+95], S. 193 ff.): Mit überschüssigen Neuronen werden solche bezeichnet, die mehrere aufeinander folgende Lernschritte (und abhängig von der Gesamtzahl der Eingabevektoren) unfixiert bleiben oder außerhalb eines zuvor festgelegten Rahmens um alle Eingabevektoren zu liegen kommen. Die Idee dabei ist, daß nach einer gewissen Trainingszeit jedes Neuron einen potentiellen Modus detektiert haben soll. Dies ist genau dann unwahrscheinlich, wenn nach dieser Zeit ein Neuron nicht in der Lage war, mehr Eingabevektoren innerhalb seiner Umgebung zu sammeln als die Gleichverteilungsschwelle vorgibt und also unfixiert blieb. Andererseits wird es für ein Neuron immer unwahrscheinlicher, einen Modus zu detektieren, wenn es mehrere hintereinander folgende Trainingszyklen von allen Eingabevektoren abgestoßen wurde und also den Eingaberaum gänzlich zu verlassen droht. Schließlich soll durch diesen Mechanismus gewährleistet werden, daß - im Sinne einer echten Vektorquantisierung - einerseits tatsächlich nur ein HRCL-Neuron einen (Sub-)Cluster beschreiben soll und andererseits alle anderen Neuronen, die nicht zur Vektorquantisierung beitragen, verworfen werden.

Im Gegensatz zum Neuronalen Gas und weiteren Competitive Learning Verfahren muß im HRCL die Anzahl der Eingabeneuronen nicht vorgegeben werden. Dies bietet den Vorteil, daß das Verfahren nicht neu gestartet werden muß, wenn dem System zu wenige Anfangsneuronen bekanntgegeben wurden (und somit nicht alle Cluster detektiert werden konnten). Die initiale Anzahl der HRCL-Neuronen wird im Top-Down Ansatz durch eine abgewandelte Zellenclustering bestimmt. Durch die Zellenclustering werden die initialen HRCL-Neuronen (auch „Seeds“ genannt) an Orten hoher Wahrscheinlichkeitsdichten plaziert. Der vorgegebene Radius r dient der Zellenclustering als Abbruchkriterium. Die Zellenclustering (s. a. [HeRo98], S. 25 ff. und [ScEr97]) unterteilt den (in jeder der n Dimensionen auf das Intervall $[-1;1]$ skalierten) Eingabektorraum in 2^n gleich große Unterräume, auch „Zellen“ genannt. Jedes der resultierenden Zellen besitzt in jeder der n Dimensionen den Radius $r_0 = 1/2$. Wenn die Anzahl der in Zelle Z_i , $i \in \{0, \dots, 2^n - 1\}$ liegenden Eingabevektoren ξ größer als die Gleichverteilungsschwelle $\Theta(r_0)$ ist, wird Zelle Z_i sukzessiv in 2^n gleich große Zellen mit Radius $r_1 = 1/4$ unterteilt, für die erneut überprüft wird, ob sie die Gleichverteilungsschwelle $\Theta(r_1)$ überschreiten. Die Prozedur wird rekursiv fortgesetzt, bis $r_j \leq r$. Die die rekursive Zellenverfeinerung bestimmende Gleichverteilungsschwelle $\Theta(r_j)$ der Verfeinerungsstufe j und skaliertes Eingabe \mathbf{D} sei dabei

$$\Theta(r_j) := \frac{(2 \cdot r_j)^n}{2^n} \cdot |\mathbf{D}| = r_j^n \cdot |\mathbf{D}| \quad (1.1)$$

$\Theta(r_j)$ gibt also die berechnete Anzahl von Eingabevektoren an, die im (angenommenen) Falle einer Gleichverteilung aller Eingabevektoren aus \mathbf{D} in Zelle der Verfeinerungsstufe j fallen. Dabei ist anzunehmen, daß Modi des multi-modalen Wahrscheinlichkeitsdichte-Raums \mathbf{D} , also Orte hoher Wahrscheinlichkeitsdichten, besonders in solchen Zellen auftreten, die mehr als $\Theta(r_j)$ Eingaben umfassen. Besitzt eine Zelle Z_j den Zellenradius $r_j \leq r$ und die Zellengrenzen $z_{\min}(d)$ und $z_{\max}(d)$ für alle Dimensionen $d \in \{1, \dots, n\}$, so wird schließlich im Zellenzentrum ein initiales HRCL-Neuron c_j plaziert mit dem Neuronengewicht bzw. -position

$$w_{c_j}(d) = z_{\min}(d) + \left(\frac{z_{\max}(d) - z_{\min}(d)}{2} \right), \quad d \in \{1, \dots, n\} \quad (1.2)$$

Die Menge \mathbf{A} der HRCL-Neuronen wird um Neuron c_j erweitert: $\mathbf{A}_{\text{neu}} = \mathbf{A}_{\text{alt}} \cup \{c_j\}$ Abb. 2.2. zeigt die Zellenclustering mit einem vom Benutzer vorgegebenen Radius $r = 0,3$ anhand der bivariaten Verteilung `modes1`. Diese enthält drei Modi und ist in jeder der Dimensionen auf das Intervall $[-1;1]$ skaliert. (Mehr zu den Eingabedaten enthält Abschnitt 3.1.) Die Verfeinerungsschritte sind für das Cluster mit Cluster-Zentroid $\mu = (0,712, -0,717)$ angedeutet.

In Abb. 2.3. sind alle aufgrund der Zellenclustering generierten 5 initialen HRCL-Neuronen für dieselbe bivariate Verteilung `modes1` und vorgegebenem Radius $r = 0,3$ gezeigt. Die 5 initialen Neuronen dienen dem nachfolgenden HRCL-Lernprozeß als Seeds.

Für wachsendes n (Radius $r \in]0, 1[$ und endlicher Eingabedatenmenge \mathbf{D}) verschwindet die Gleichverteilungsschwelle $\Theta(r)$ und es gilt folgende Gleichung:

$$\lim_{n \rightarrow \infty} \Theta(r) = \lim_{n \rightarrow \infty} r^n \cdot |\mathbf{D}| = 0 \quad (1.3)$$

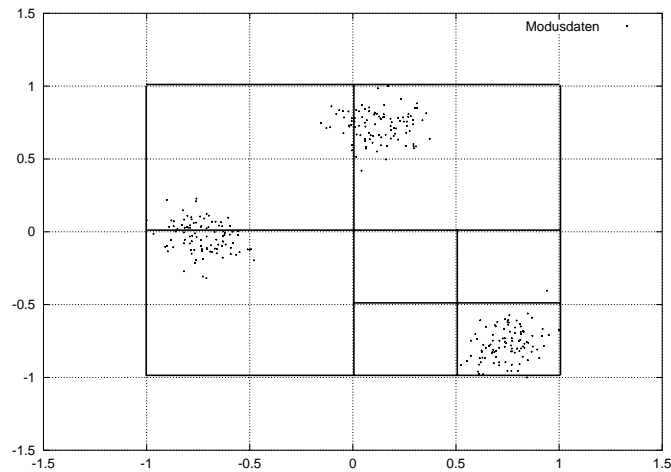


Abb. 2.2.: Zellenverfeinerung mit Radius $r = 0,3$ anhand der bivariaten Verteilung `modus1`

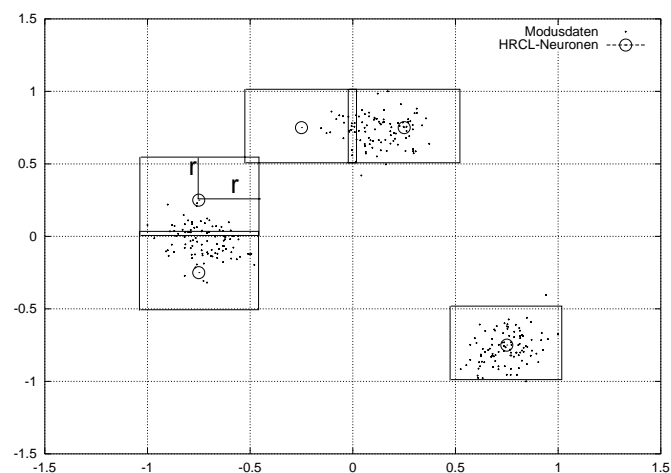


Abb. 2.3.: Automatische Generierung von 5 initialen HRCL-Neuronen. Gezeigt sind die HRCL-Neuronen c_i inklusive ihren durch Radius $r = 0,3$ definierten Umgebungen $U_{c_i}(r)$ für die bivariate Verteilung `modus1`

Aus diesem Grund wird für höherdimensionale Eingaberäume $\Theta(r)$ durch den Median der $|U_c(r)|$ (d. h. der Anzahl der durch Umgebung $U_c(r)$ eingeschlossenen Eingaben ξ) aller durch die Zellenclustering generierten HRCL-Neuronen c angenähert¹. Falls die Umgebungsgrößen $|U_{c_i}(r)|$ für alle c_i der Größe nach sortiert sind gilt also:

1. Für sehr hoch-dimensionale Eingaberäume kann auf die rechenintensive Zellenclustering verzichtet werden und $\Theta(r)$ effizienter durch den Median der $|U_c(r)|$ von $N = \sqrt{|\mathbf{D}|}$ HRCL-Neuronen bestimmt werden, die zufällig auf Eingaben ξ aus \mathbf{D} gesetzt werden. Dies entspricht im wesentlichen dem 1. Schritt des Bottom-Up HRCL (s. a. Abschnitt 2.2.).

$$\Theta(r) \equiv |U_{c_m}(r)|, \quad m = \frac{N}{2} \quad (1.4)$$

Nachdem durch die Zellenclustering alle initialen HRCL-Neuronen bestimmt sind und mindestens ein Neuron erzeugt wurde, adaptiert das HRCL die ihm zur Verfügung stehenden Neuronen. Das HRCL-Lernverfahren soll in Abschnitt 2.1. im Detail beschrieben werden.

2.1. Top-Down HRCL

1. *Schritt:* Skaliere die Eingabedatenmenge \mathbf{D} in jeder Dimension $d \in \{1, \dots, n\}$ auf das Intervall $[-1;1]$. Bestimme die maximale Anzahl von HRCL-Lernschritten abhängig von der Anzahl der Eingabevektoren: $t_{\max} := \lambda_t \cdot |\mathbf{D}|$ (Für λ_t verwendet man sehr oft den heuristischen Wert 8). Initialisiere den Zeitparameter $t := 0$.
2. *Schritt:* Wähle zufällig ein Eingabedatum ξ aus der Eingabe-Datenmenge \mathbf{D} aus. Setze die Anzahl der noch unadaptierten HRCL-Neuronen $\lambda_2 := 0$.
3. *Schritt:* Bringe alle HRCL-Neuronen aus \mathbf{A} abhängig von den Distanzen ihrer Referenzvektoren w_{c_i} zur Eingabe ξ in eine Reihung (z. B. mit Hilfe von Quicksort; [Sedg92], S. 115 ff.). Der Index $k_i(\xi, \mathbf{A})$ soll dabei - analog zum Neuronalen Gas - die Ordnung von w_{c_i} wiedergeben, angefangen bei $k_i(\xi, \mathbf{A}) = 0$, falls c_i der Gewinner ist, bis $k_i(\xi, \mathbf{A}) = 1$ für dasjenige Neuron, das von der aktuellen Eingabe ξ am weitesten entfernt ist.
4. *Schritt:* Berechne die neue Position von Neuron c_i , $i \in \{1, \dots, N\}$ mit Hilfe der Adaptionregel

$$w_i(t+1) = w_i(t) + f_{\lambda_1, \lambda_2}(k_i(\xi, \mathbf{A})) \cdot g_{\lambda_3, \lambda_4}(t) \cdot (\xi - w_i) \quad (1.5)$$

mit

$$f_{\lambda_1, \lambda_2}(k_i(\xi, \mathbf{A})) := e^{\frac{-\lambda_1 \cdot (k_i(\xi, \mathbf{A}) - \lambda_2)}{N}} \quad (1.6)$$

(Dabei sei $-\lambda_1$ die Rate des Abfalls an Adaption, gemessen vom Gewinner bis entferntesten Neuron, d. h. von $k_i(\xi, \mathbf{A}) = 0$ bis $k_i(\xi, \mathbf{A}) = 1$. λ_2 sei die Anzahl der noch unadaptierten HRCL-Neuronen. N sei die Gesamtzahl an Neuronen der Neuronenmenge \mathbf{A} .)

sowie

$$g_{\lambda_3, \lambda_4}(t) := \frac{1}{\lambda_3} e^{-\lambda_4 \frac{t}{t_{\max}}} \quad (1.7)$$

(λ_3 definiere die Anfangsadaptionsrate zum Zeitpunkt $t = t_0$, $-\lambda_4$ definiere den Abfall der Adaption über die Zeit t .)

f drückt dabei das Maß an Adaption bezüglich aller Neuronen der geordneten Neuronenreihe aus, wohingegen g mit der Zeit t gegen 0 strebt. f bringt zum Ausdruck, daß der Gewinner am stärksten adaptiert werden soll, das von ξ entfernteste Neuron am geringsten. Die Funktion g soll schließlich das HRCL-Lernverfahren mit anwachsender Zeit stabilisieren.

5. *Schritt:* Berechne die neue Position von Neuron c_i , $i \in \{1, \dots, N\}$ für den Fall, daß c_i von einem bereits adaptierten Neuron abgestoßen wird mit Hilfe folgender Abstoßregel

$$w_i(t+1) = w_i(t) - f_{\lambda_1, \lambda_2}(k_i(\xi, \mathbf{A})) \cdot g_{\lambda_3, \lambda_4}(t) \cdot (\xi - w_i) \quad (1.8)$$

mit den Abstoßfunktionen

$$f_{\lambda_1, \lambda_2}(k_i(\xi, \mathbf{A})) := \left(\frac{1}{\lambda_1} \cdot -e^{\frac{\lambda_2 \cdot k_i(\xi, \mathbf{A})}{N}} \right) + 1 \quad (1.9)$$

und

$$g_{\lambda_3, \lambda_4}(t) := e^{\lambda_3 \frac{t}{t_{\max}}} - \lambda_4 \quad (1.10)$$

Wie im Fall der Neuronenadaption definieren λ_1 und λ_2 die Abfallrate der Abstoßfunktion f gemessen über den Neuronenindex $k_i(\xi, \mathbf{A})$, λ_3 die Anstiegsrate der Abstoßfunktion g und λ_4 die Abstoßrate eines beliebigen Neurons zum initialen Zeitpunkt $t = t_0$. Die Abstoßfunktion f nimmt dabei mit der Ordnung der HRCL-Neuronen ab, d. h. das dem Gewinner nächste Neuron wird maximal von der konkurrierenden Eingabe ξ abgestoßen, das von ihm am weitesten entfernte wird nur noch gering von ξ entfernt. Dabei wird die Rate des Abstoßens eines Neurons von einer Eingabe - ausgehend von einer minimalen initialen Abstoßrate λ_4 - über der Zeit t maximiert.

Abb. 2.4. (a) zeigt die Adaptionsrate bezüglich der Neuronenordnung $k_i(\xi, \mathbf{A})$ und Parameter $-\lambda_1 = -5,5$. Dabei habe der Gewinner Ordnungsindex $k_i(\xi, \mathbf{A}) = 0$ und das am weitesten von ξ entfernte Neuron c_i den Index $k_i(\xi, \mathbf{A}) = 1$. Abb. 2.4. (b) zeigt die Adaptionsrate gemessen über die Zeit t von $t = t_0 := 0$ bis $t = t_{\max} := 1$ und Adaptions-Parametern $\lambda_4 = 10$ und $\lambda_3 = 1,25$. Die Adaption der Neuronen nimmt nicht nur mit der Distanz der Neuronen zur Eingabe ab, sondern auch über die gesamte Lernzeit t . Ebenso werden Neuronen je kräftiger von einer Eingabe abgestoßen, desto näher sie zur Eingabe zu liegen kommen und je kräftiger abgestoßen desto mehr Lernschritte t verstrichen sind (s. Abb. 2.5.). Dabei liegt die minimale Abstoßrate von g zum Zeitpunkt $t = 0$ bei Abstoß-Parametern $\lambda_3 = 10$ und $\lambda_4 = 0,9$ bei 0,1 und die maximale Abstoßrate bei 30 % der maximal zur Verfügung stehenden Lernzeit t bereits bei (theoretischen) 1.900 %. Das HRCL-Lernverfahren nutzt nicht die maximal zur Verfügung stehende Lernzeit t_{\max} , sondern bricht das Lernen in der Regel schon sehr viel früher ab (s. 10. Schritt).

6. *Schritt:* Adaptiere das Gewinnerneuron s_w mit der in Gl. (1.5) angegebenen Adaptionsregel, falls $|U_{s_w(t+1)}(r)| \geq |U_{s_w(t)}(r)|$ oder falls s_w unfixiert ist. Unfixiere den zweiten Gewinner der Neuronenordnung, für den gilt $k_i(\xi, \mathbf{A}) = 1/N$, für den Fall, daß sich die beiden Neuro-

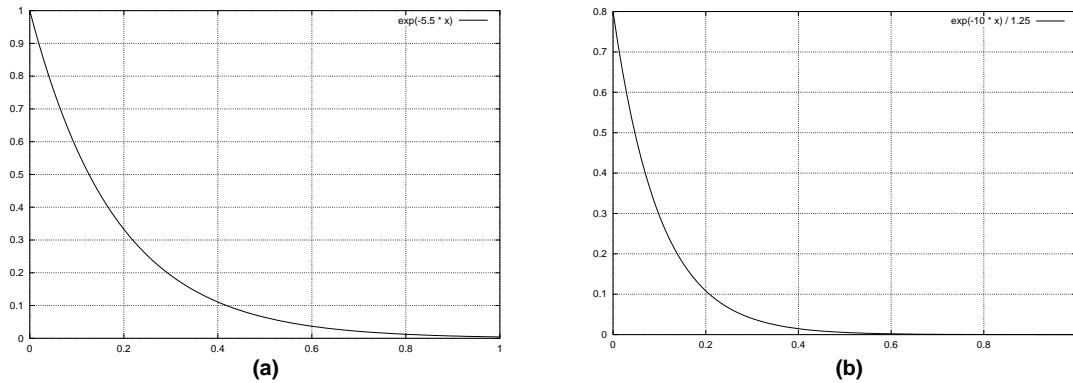


Abb. 2.4.: (a) Adaptionrate relativ zur Neuronenordnung $k_i(\xi, \mathbf{A})$ und
(b) Adaptionrate der Neuronen über die Lernzeit t von $t = 0$ bis
 $t = t_{\max} := 1$.

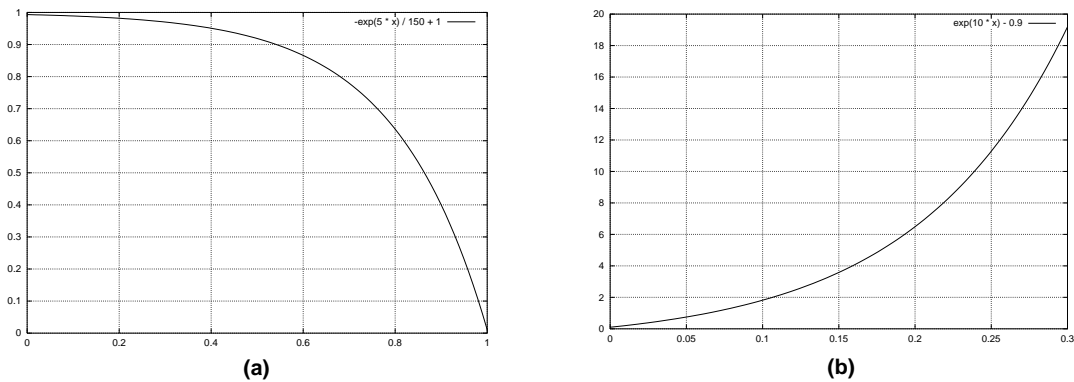


Abb. 2.5.: Abstoßrate (a) relativ zur Neuronenordnung und
(b) relativ zur Lernzeit t (dargestellt für den Zeitraum von $t = 0$ bis
 $t = 30\% \cdot t_{\max}$).

nen-Umgebungen von Gewinner und zweitem Gewinner in einer der n Dimensionen über ein zuvor festgelegtes „Überlappungsmaß“ überschneiden (s. auch 7. Schritt). Fixiere das Gewinnerneuron s_w , falls $|U_{s_w(t+1)}(r)| \geq \Theta(r)$.

7. Schritt: Für alle anderen Neuronen c_i , $c_i \neq s_w$ der Neuronenfolge gilt:

7.1.1. Falls sich die beiden Umgebungen $U_{s_w(t+1)}(r)$ und $U_{c_i(t+1)}(r)$, ($c_i \neq s_w$) in einer der n Dimensionen über ein zuvor festgelegtes „Überlappungsmaß“ überschneiden und c_i unfixiert ist, so stoße c_i mit der in Gl. (1.8) angegebenen Abstoßrate von der Eingabe ξ ab. Fixiere, falls notwendig, das Neuron c_i in der neuen Position $w_i(t+1)$.

7.1.2. Falls sich die beiden Umgebungen des Gewinners und des Neurons c_i in unerlaubter Weise überlappen und c_i fixiert ist, so behält c_i seine alte Position und wird nicht adaptiert, also: $w_i(t+1) = w_i(t)$. Erhöhe gleichzeitig die Anzahl der noch nicht adaptierten Neuronen λ_2 aus Schritt 2 um 1.

2. HRCL

7.2.1. Falls es zu keinerlei Überlappung zwischen Gewinner und c_j kommt, so adaptiere c_j mit Hilfe von Gl. (1.5), falls $|U_{c_i(t+1)}(r)| \geq |U_{c_i(t)}(r)|$ oder falls c_j unfixiert ist.

7.2.2. Andernfalls behalte Neuron c_j seine alte Position, also $w_j(t+1) = w_j(t)$ und $\lambda_2 := \lambda_2 + 1$ (s. o.).

8. *Schritt:* Erhöhe Zeitparameter t um 1: $t := t + 1$.
9. *Schritt:* Verwerfe alle solche Neuronen $c_j \in \mathbf{A}$, die $\lambda_{\text{prune}}(|\mathbf{D}|)$ aufeinander folgende Lernschritte unfixiert bleiben oder welche in einer der n Dimensionen außerhalb des Intervalls $[-1,5;1,5]$ zu liegen kommen („Neuronen-Pruning“). Der Pruningparameter wird in der Regel auf $\lambda_{\text{prune}}(|\mathbf{D}|) = 1/15 \cdot |\mathbf{D}|$ gesetzt.
10. *Schritt:* Beende das HRCL-Lernen, falls die erste Ableitung des erwarteten Quantisierungsfehler EQE $\lambda_{\text{eqe}}(|\mathbf{D}|)$ aufeinander folgende Lernschritte annähernd den Wert 0 annimmt, d. h. falls sich der erwartete Quantisierungsfehler über eine längere Zeit nicht wesentlich ändert (s. a. Gl. (1.12), S. 14). Der Quantisierungsparameter $\lambda_{\text{eqe}}(|\mathbf{D}|)$ ist in Analogie zu $\lambda_{\text{prune}}(|\mathbf{D}|)$ von der Anzahl der Eingabevektoren abhängig und wird i. d. R. mit dem Wert $\lambda_{\text{eqe}}(|\mathbf{D}|) = 1/20 \cdot |\mathbf{D}|$ belegt.
11. *Schritt:* Verwerfe die verbliebenen Neuronen nach der Vorschrift in Schritt 9 („Neuronen-Pruning“).
12. *Schritt (Hierarchische Verfeinerung):* Falls die Anzahl der resultierenden HRCL-Neuronen $N > 1$, so setze $\mathbf{D} := \{\xi | \xi \in U_{c_i}(r)\}$, $i \in \{1, \dots, N\}$, skaliere die neue Eingabedatenmenge \mathbf{D} in jeder Dimension $d \in \{1, \dots, n\}$ auf das Intervall $[-1;1]$, bestimme die maximale Anzahl von HRCL-Lernschritten abhängig von der Anzahl der Eingabevektoren: $t_{\text{max}} := \hat{\lambda}_t \cdot |\mathbf{D}|$ (s. 1. Schritt) und generieren N initiale HRCL-Neuronen mit Hilfe der zuvor beschriebenen Zellenclustering. Gehe dann zur nächsten Hierarchiestufe, indem das Programm bei Schritt 2 wieder aufgenommen wird.

Im folgenden soll der zentrale 7. Schritt des HRCL-Lernverfahrens graphisch veranschaulicht werden. Im Fall einer Überlappung der Umgebungen zwischen Gewinner s_w und zweitem Gewinner c_j wird der zweite Gewinner von der konkurrierenden aktuellen Eingabe ξ abgestoßen (Schritt 7.1.1.; s. Abb. 2.6.). Eingezeichnet ist die verbotene Überlappung der beiden Umgebungen $U_{s_w(t+1)}(r)$ und $U_{c_i(t+1)}(r)$, für den Fall daß c_j regulär adaptiert werden würde, mit einem angenommenen Radius $r = 0,1$ (unterbrochene Pfeillinie), sowie die tatsächliche Adaption bzw. Abstoßung der beiden Neuronen (durchgezogene Pfeile). $c_j(t)$ ist dabei unfixiert. Wäre dagegen $c_j(t)$ fixiert, so würde c_j seine alte Position behalten und nicht von ξ abgestoßen werden (Schritt 7.1.2.). Die Fälle 7.2.1. und 7.2.2. verlaufen analog zu den obigen Fällen und werden an dieser Stelle nicht gezeigt.

Insgesamt minimiert das HRCL Lernverfahren die Anzahl der Neuronen, bis gerade noch so viele Neuronen übrig bleiben, um jede Wahrscheinlichkeitsdichte oder Modus durch genau ein HRCL-Neuron repräsentieren zu können. Ein für die Vektorquantisierung optimaler Zustand wird dabei erreicht, wenn die Neuronen auf die Modus-Zentren plaziert werden. Zugleich versucht das Verfahren den erwarteten Quantisierungsfehler EQE zwischen dem Verwerfen (Pruning) zweier HRCL-Neuronen zu minimieren.

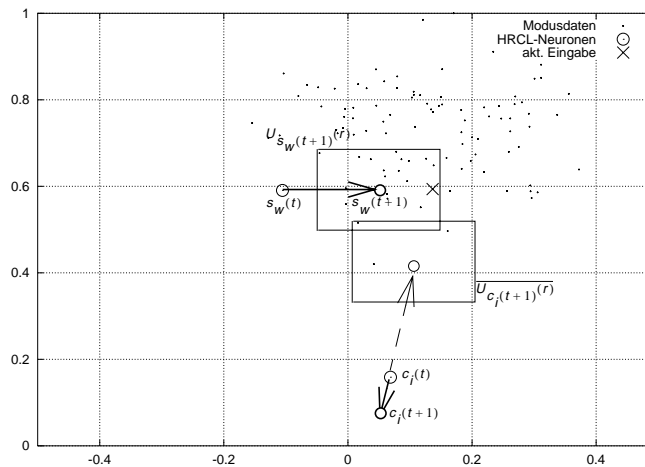


Abb. 2.6.: Abstoßen des zweiten Gewinners c_j von der aktuellen Eingabe ξ wegen Überlappung mit Gewinnerneuron s_w .

Nach jedem Neuronen-Pruning steigt der erwartete Quantisierungsfehler wieder an (s. a. Gl. (1.12) und (1.14)). Das HRCL-Training stabilisiert sich schließlich, falls keine Neuronen mehr verworfen werden, der erwartete Quantisierungsfehler sich über einen längeren Zeitraum nur noch geringfügig ändert und also das Verfahren für die finalen HRCL-Neuronen ein lokales Fehlerminimum gefunden hat. In diesem Zustand wird das HRCL-Training beendet (s. 10. Schritt).

In Abschnitt 3.4. werden Ergebnisse des Top-Down HRCL gezeigt, die mit verschiedenen bivariaten Eingaben erzielt werden können. Vor allem wird auch auf die Verwendung des Top-Down HRCL bei der Detektierung globulärer Cluster eingegangen.

2.2. Bottom-Up HRCL

Da das Top-Down HRCL von etlichen Parametern, vor allem aber vom Benutzer vorzugebenden Radius r , abhängig ist, wird im folgenden das Top-Down HRCL durch das Bottom-Up HRCL ergänzt. Das Bottom-Up HRCL verfeinert nicht wie das Top-Down Verfahren eine grobe Ansicht der ersten Hierarchiestufe, um potentielle Subcluster zu erkennen, sondern, versucht umgekehrt in der ersten Hierarchiestufe kleinste Wahrscheinlichkeitsdichten im Eingabedatensatz zu erkennen. Diese werden in nachfolgenden Hierarchiestufen sinnvoll zu immer größeren Wahrscheinlichkeitsdichten zusammengefaßt. Beiden Varianten des HRCL nehmen dabei vereinfachend an, daß der Eingeraum aus normalverteilten Modi besteht, und daß infolgedessen Wahrscheinlichkeitsdichten des Eingaberaums durch Umgebungen mit festem Radius approximiert werden können. Diese grobe Vereinfachung der tatsächlichen Bedingungen versucht der Top-Down Ansatz zu beseitigen, indem in nachfolgenden Hierarchiestufen Details der nur groben detektierten Cluster erkannt werden sollen, wohingegen das Bottom-Up HRCL in nachfolgenden Hierarchiestufen Cluster konstruiert, die sich aus mehreren kleineren Normalverteilungen zusammensetzen sollen.

Im Gegensatz zum Top-Down Ansatz benötigt das Bottom-Up HRCL keine Zellenclusterung, um initiale Neuronen zu plazieren. Die Seeds werden stattdessen gleich über die Eingaben verteilt. Zudem wird für alle Hierarchiestufen derselbe Eingabedatensatz benutzt. Der Eingabedatenmenge muß also nicht vor jeder Hierarchiestufe skaliert werden, was u. U. zu problematischen Verzerrungen des Eingabedatensatzes führen kann.

3. Vergleichende Ergebnisse

In der folgenden Beschreibung sollen vor allem die Unterschiede zwischen Top-Down und Bottom-Up HRCL herausgearbeitet werden. Für den größten Teil des Verfahrens kann auf den Top-Down Algorithmus in Abschnitt 2.1. verwiesen werden, da die beide Varianten viele Gemeinsamkeiten haben und sich letztendlich nur in der Art der Hierarchisierung unterscheiden.

1. *Schritt*: Skalierung und Initialisierung wie in der Top-Down Variante. Generiere zudem $N := \sqrt{|\mathbf{D}|}$ Neuronen c_i so, daß sie zufällig mit Eingaben ξ aus der Eingabedatenmenge \mathbf{D} zusammenfallen und die Bedingung erfüllen, daß ihre Umgebungen $U_{c_i}(r)$ sich nicht in unerlaubter Weise überlappen.
2. *bis 11. Schritt*: analog zu Top-Down
12. *Schritt (Hierarchische Verfeinerung)*: Falls die Anzahl der resultierenden HRCL-Neuronen $N > 1$ ist, erhöhe den Radius $r := \lambda_{\text{radius}} \cdot r$ (λ_{radius} ist i. d. R. mit 2 vorgegeben.), berechne ausgehend vom neuen Radius eine neue Gleichverteilungsschwelle $\Theta(r)$, erzeuge \sqrt{N} initiale Neuronen analog zum Schritt 1 und beginne mit der nächsten Hierarchiestufe, indem das Programm an Schritt 2 fortgesetzt wird.

3. Vergleichende Ergebnisse

Um die beiden HRCL-Varianten mit anderen (statistischen und neuronalen) Verfahren vergleichen zu können, müssen objektive Beurteilungskriterien gefunden werden, die die Güte des Clusterverfahren einzuschätzen versuchen. Sowohl für statistische als auch neuronale Verfahren haben sich zwei Maßstäbe etabliert: der erwartete Quantisierungs- bzw. Verzerrungsfehler („Expected Quantisation Error“, EQE oder „Expected Distortion Error“) sowie die Entropie. Der zu minimierende *Quantisierungsfehler* ist dabei für kontinuierliche Signalverteilungen $p(\xi)$

$$EQE(p(\xi), \mathbf{A}) = \sum_{c \in \mathbf{A}} \int_{V_c} \|\xi - w_c\|^2 p(\xi) d\xi \quad (1.11)$$

mit der Voronoi-Umgebung V_c um Neuron $c \in \mathbf{A}$ und im Fall einer diskreten und endlichen Eingabedatenmenge \mathbf{D} und Voronoi-Menge R_c um Neuron c (s. a. [HeRo98], S. 30 ff. und S. 27)

$$EQE(\mathbf{D}, \mathbf{A}) = \frac{1}{|\mathbf{D}|} \sum_{c \in \mathbf{A}} \sum_{\xi \in R_c} \|\xi - w_c\|^2 \quad (1.12)$$

Die Voronoi-Menge R_c um Neuron c ist definiert als die Menge aller Eingaben ξ , die näher zu Neuron c (mit Gewicht w_c) liegen als zu jedem anderen Neuron der Neuronenmenge $\mathbf{A} = \{c_1, c_2, \dots, c_N\}$, d. h.

$$R_c := \{\xi \mid \|\xi - w_c\| < \|\xi - w_i\| \quad \forall i \neq c\} \quad (1.13)$$

Der Quantisierungsfehler EQE mißt den Abstand aller Eingaben ξ von den Referenz- oder Gewichtsvektor w_c ihrer nächsten Neuronen c und ist besonders gering bei geclusterten Eingabedatenmengen \mathbf{D} . Ein triviales Optimum $EQE(\mathbf{D}, \mathbf{A}) = 0$ wird natürlicherweise erreicht, wenn alle Neuronen bzw. ihre Gewichte oder Positionen w_c mit den Eingaben ξ zusammenfallen. Daraus läßt sich umgekehrt schließen, daß ein Neuronen-Pruning den Quantisierungsfehler erhöht. Um für diesen Fall trotzdem einen aussagekräftigen Quantisierungsfehler anzugeben, wird der Quantisierungsfehler EQE durch einen normierten Quantisierungsfehler NQE ersetzt (vgl. auch „F-Kriterium“: [HeHe95], S. 57 ff.). Beim normierten Quantisierungsfehler NQE geht die Anzahl der HRCL-Neuronen in die Berechnung des Quantisierungsfehlers ein, der sich wie folgt berechnet:

$$\begin{aligned} NQE(\mathbf{D}, \mathbf{A}) &= \left(1 - \frac{|\mathbf{D}| - |\mathbf{A}|}{|\mathbf{D}|}\right) \cdot EQE(\mathbf{D}, \mathbf{A}) \\ &= \frac{|\mathbf{A}|}{|\mathbf{D}|^2} \cdot \sum_{c \in \mathbf{A}} \sum_{\xi \in R_c} \|\xi - w_c\|^2 \end{aligned} \quad (1.14)$$

Der Normierungsterm in Gl. (1.14)

$$1 - \frac{|\mathbf{D}| - |\mathbf{A}|}{|\mathbf{D}|} \quad (1.15)$$

schwankt dabei zwischen zwei extremen Werten und ist 0, falls keine HRCL-Neuronen vorhanden sind und ist 1, falls soviele Neuronen wie Eingabevektoren vorhanden sind. Der normierte Quantisierungsfehler $NQE(\mathbf{D}, \mathbf{A})$ stellt ein Gleichgewicht zwischen Quantisierungsfehler und Anzahl der HRCL-Neuronen her und wächst also mit $EQE(\mathbf{D}, \mathbf{A})$ und mit der Anzahl der verworfenen („geprunten“) Neuronen an und wird gleichzeitig durch den Normierungsterm aus Gl. (1.15) verkleinert.

Ein mit der Vektorquantisierung konkurrierendes Ziel verfolgt die Maximierung der *Entropie*: Die Entropie ist genau dann am größten, wenn die Neuronen bzw. ihre Referenzvektoren so im Eingaberaum, der durch die Eingabedatenmenge \mathbf{D} beschrieben wird, verteilt sind, daß jedes Neuron c innerhalb seiner Voronoi-Umgebung V_c die gleiche Menge an Eingaben umfaßt. Ist maximale Entropie erreicht, so besitzt jedes Neuron die gleiche Chance Gewinner einer zufällig erzeugten Eingabe ξ aus \mathbf{D} zu sein (s. a. [HeRo98], S. 31). Für die Entropy ENT und den kontinuierlichen Fall gilt also, daß

$$\int_{V_c} p(\xi) d\xi = \frac{1}{|\mathbf{A}|} \quad (\forall c \in \mathbf{A}) \quad (1.16)$$

der für den diskreten und endlichen Fall in die Gleichung

$$\frac{|R_c|}{|\mathbf{D}|} = \frac{1}{|\mathbf{A}|} \quad (\forall c \in \mathbf{A}) \quad (1.17)$$

3. Vergleichende Ergebnisse

übergeht (vgl. [Fritz97], S. 8). Gl. (1.17) drückt aus, daß im Fall maximaler Entropie der relative Anteil der in der Voronoi-Menge R_c um Neuron c eingeschlossenen Eingabevektoren ξ gleich dem prozentualen Anteil desselben Neurons c an der Neuronenmenge \mathbf{A} ist. Dies muß für alle Neuronen c aus \mathbf{A} gelten.

Um maximale Entropie zu erreichen, genügt es also

$$ENT(\mathbf{D}, \mathbf{A}) = \sum_{c \in \mathbf{A}} \left\| 1 - \frac{|R_c| |\mathbf{A}|}{|\mathbf{D}|} \right\| \quad (1.18)$$

zu minimieren.

Um einen objektiven Vergleich zwischen verschiedenen Verfahren zu ermöglichen, müssen nicht nur die obenn genannten objektiven Beurteilungskriterien festgelegt werden, sondern die durchzuführenden Experimente müssen zudem auf einheitlichen bzw. künstlich erzeugten Eingabedatensätzen erfolgen. Im weiteren Abschnitt werden die verwendeten Eingabedatenmengen \mathbf{D} vorgestellt. Jede einzelne Datensatz besteht aus einer endlichen Anzahl von Vektoren, die durch eine diskrete Wahrscheinlichkeitsdichte-Funktion $\rho(\xi)$, $\xi \in \mathbf{R}^n$ erzeugt werden, welche eine zwei-dimensional bzw. bivariat verteilte multimodale Signalverteilung repräsentiert. Zweidimensionale Eingabedaten bieten im Gegensatz zu vier- oder höherdimensionalen Eingaben zudem den Vorteil, daß Ergebnisse der Neuronenadaption visualisiert werden können: Oftmals können nämlich scheinbar objektive Beurteilungskriterien wie der erwartete Quantisierungsfehler nicht die tatsächliche Qualität des Clusterverfahrens widerspiegeln (s. a. Gl. (1.14)). Dieser Defekt läßt sich leicht aus den tatsächlichen 2-dimensionalen Adaptionsergebnissen folgern (vgl. hierzu auch Abschnitt 3.3. und 3.5.).

Alle Experimente werden auf einer Sun Sparc Ultra 2 (2 CPUs, 640 MByte Hauptspeicher) durchgeführt, um insbesondere einheitliche Untersuchungen des Laufzeitverhaltens durchführen zu können.

3.1. Eingabedaten

Um die künstlichen Eingabedatenmengen `modes1`, `modes9` und `modes12` zu erzeugen, werden für jeden Modus $M' := |\mathbf{D}| / m$ n -dimensionale gaußverteilte Zufallszahlen

$$t_n = \mu_{m,n} + \sigma_{m,n} \left(\sum_{j=1}^{12} \text{rand}_j(1) - 6 \right) \quad (1.19)$$

berechnet. Dabei gibt m die Anzahl der im Datensatz vorhandenen Modi an, $\mu_{m,n}$ den n -dimensionalen Mittelwert oder Moduszentrum des m -ten Modus', $\sigma_{m,n}$ die Varianz des m -ten Modus' in der n -ten Dimension und $\text{rand}_j(1)$ eine j -te Zufallszahl zwischen 0 und 1 (vgl. [Hamm73]).

Der Datensatz `modes1` besitzt folgende Parameter: $m = 3$, $n = 2$, $M' = 100$, $\mu_0 = (1, 7)^t$, $\sigma_0 = (1, 1)^t$, $\mu_1 = (-7, 0)^t$, $\sigma_1 = (1, 1)^t$, $\mu_2 = (7, -7)^t$ und $\sigma_2 = (1, 1)^t$ und besteht aus drei gaußverteilten Modi mit Varianzen 1 in jeder der 2 Dimensionen. `Modes1` besteht also insgesamt aus 300 2-dimensionale Vektoren.

Der Eingabedatensatz `modes9` besteht aus 3 globulären Modi. Jedes dieser Modi besteht aus 5 Submodi und einem zentrierten, mit der 10-fachen Varianz überlagerten gaußverteilten Rauschen. Der erste globuläre Modus wird durch folgende Parameter beschrieben: $m = 6$, $n = 2$ und $M' = 100$ sowie durch die 6 Modusmittelwerte und -varianzen: $((0, 0)^t, (1, 1)^t)$, $((6, 6)^t, (1, 1)^t)$, $((-6, -6)^t, (1, 1)^t)$, $((-6, 6)^t, (1, 1)^t)$, $((6, -6)^t, (1, 1)^t)$, $((0, 0)^t, (10, 10)^t)$. Der zweite globuläre Modus ist identisch zum ersten Modus mit jeweils um $(35, 35)$ translatierten Mittelwerten. Der dritte globuläre Modus ist um $(41, -6)$

bezüglich des ersten translatiert und enthält leichte Variationen in den Mittelwerten der Submodi (s. a. Abb. 2.7. (a)). `modus9` enthält insgesamt 1.800 2-dimensionale Vektoren.

`modus12` ist ein „irregulärer“ Datensatz, der aus verschiedenen, teilweise überlappenden Modi unterschiedlicher Varianzen in unterschiedlichen Dimensionen besteht und durch folgende Werte charakterisiert wird: $m = 6$, $n = 2$, $M = 100$, sowie Modusmittelwerte und -varianzen: $((-15, -8)^t, (1, 1)^t)$, $((-11, -2)^t, (4, 3)^t)$, $((8, 9)^t, (1, 1)^t)$, $((11, 12)^t, (1, 1)^t)$, $((8, -8)^t, (1, 6)^t)$, $((0, 0)^t, (1, 12)^t)$, $((0, 0)^t, (12, 12)^t)$. In ihm sind drei Modi mit einfacher Varianz in jeder der zwei Dimensionen, drei Modi mit unterschiedlichen Varianzen in den beiden Dimensionen und ein mit der zwölffachen Varianz überlagerter Modus als Rauschen angeordnet. `modus12` enthält damit insgesamt 600 2-dimensionale Eingabevektoren plus 100 2-dimensionale Vektoren, die ein gaußverteiltes Rauschen darstellen sollen.

Die Eingabedaten werden auf das reelle Intervall $[-1;1]$ in jeder der n Dimensionen skaliert. `modus1` ist in Abb. 2.2. und Abb. 2.3., die Eingabedatensätze `modus9` und `modus12` sind in Abb. 2.7. (a) bzw. Abb. 2.7. (b) dargestellt.

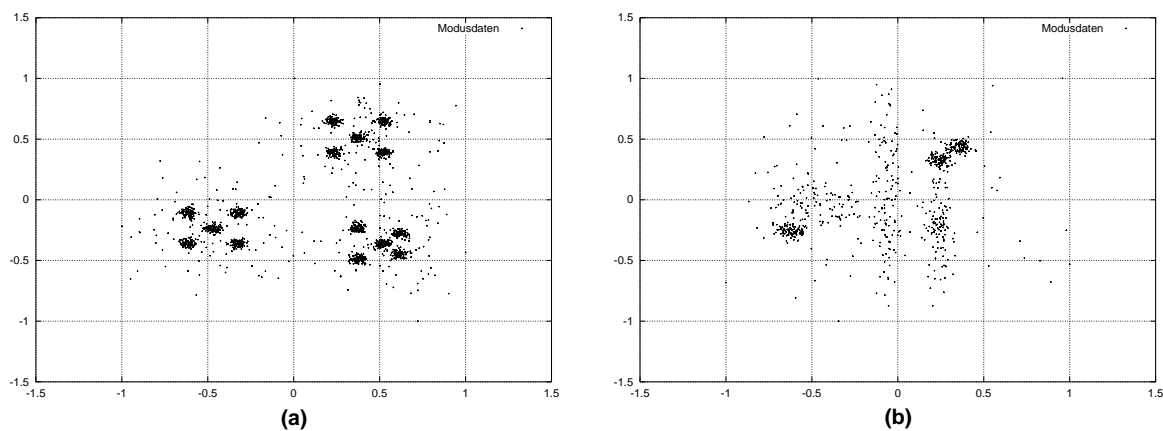


Abb. 2.7.: (a) `modus9` und (b) `modus12` Eingabedatenmengen

Weitere Eingabedatensätze können auf die gleiche Art erzeugt werden.

3.2. SMART

Die Leistungsfähigkeit des SMART-Retrievals soll in diesem Abschnitt stellvertretend für weitere statistische Single-Pass Clusterverfahren untersucht werden. Das SMART-Retrieval Verfahren verarbeitet die Eingaben seriell und ordnet dem ersten Eingabedatum den ersten Clusterprototypen bzw. -zentroid zu. Alle weiteren Eingaben werden dem ersten Clusterprototypen entweder zugewiesen, falls sie innerhalb der Schwelle r um diesen Prototypen liegen oder erzeugen einen neuen Prototypen. Nachfolgende Eingaben werden auf die gleiche Art mit allen verfügbaren Clusterprototypen verglichen. Überlappungen der Schwellen r verschiedener Prototypen sind erlaubt, wenn sie durch den Benutzer vorgegeben werden. (s. [HeRo98], S. 15). Zwei exemplarische Ergebnisse des SMART-Retrievals mit der benutzerdefinierten Schwelle $r = 0,3$ und einer Überlappung von 0 % sind in Abb. 2.8. wiedergegeben. Das Verfahren ist dabei, wie in [SaMc83] bereits angedeutet, stark von der Reihenfolge der zu verarbeitenden Eingabevektoren bzw. Dokumente abhängig und reagiert zudem deutlich auf „Ausreißer“ im Datensatz, indem Clusterprototypen auch an Eingaben plaziert werden, die am Rand eines Clusters liegen (vgl. Abb. 2.8.). Es werden 10 voneinander unabhängige Läufe realisiert und deren Ergebnisse dokumentiert:

Der Quantisierungsfehler EQE nimmt während der Adaption an den Eingabedatensatz `modus1` von

3. Vergleichende Ergebnisse

einem Startwert von $\varnothing 1,6 \pm 0,5$ bis zu einem Wert von $\varnothing 0,023 \pm 0,003$ ab. Wie insbesondere Abb. B.1. rechts in Abschnitt B.1. zeigt, kann sich der EQE jedoch über die gesamte Adaptionzeit nicht stabilisieren. Deutlich schlechter verhält sich die Entropie, was erneut darauf hinweisen mag, daß SMART empfindlich auf Ausreißer reagiert: Die Entropie wird durch das SMART-System nicht maximiert, sondern sogar deutlich verkleinert (d. h. ENT wächst deutlich von 0 auf $\varnothing 5,75 \pm 1,75$ an; s. Abb. B.2.). Die Laufzeit beträgt ca. 1 min 25 sec \pm 7 sec.

Untersuchungen anhand weiterer Eingabedatensätze liefern keine wesentlich anderen Ergebnisse und werden deshalb nicht dargestellt.

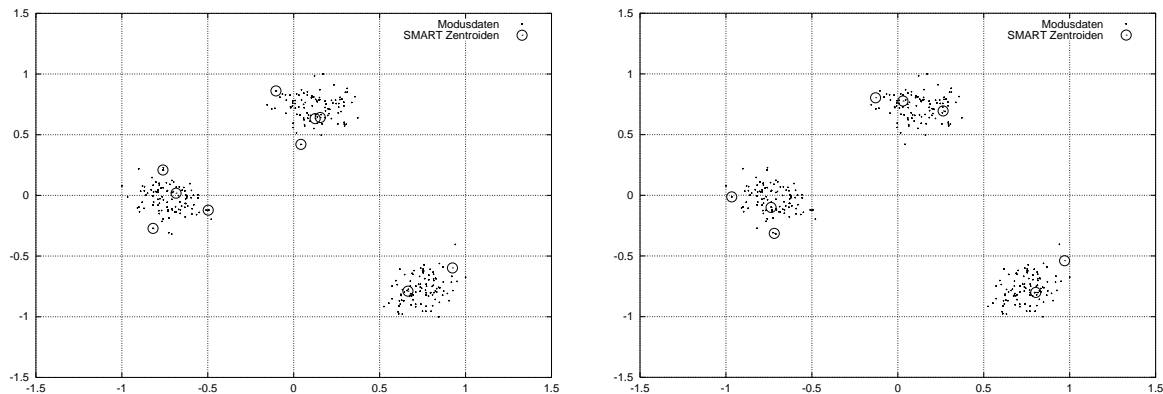


Abb. 2.8.: Zwei exemplarische Ergebnisse des SMART-Retrievals bezüglich der Eingabe `modes1` nach je 300 Adaptionsschritten und $r = 0,3$.

3.3. SOM

Kohonen Selbst-Organisierende Karte (SOM) ist der bekannteste Vertreter der Soft Competitive Learning Verfahren mit vorher festgelegter Dimensionalität der Ausgabeschicht (s. [HeRo98], S. 33 ff.). Dem Standard SOM-Algorithmus müssen vor dem Training sowohl die Anzahl der Neuronen, deren Anordnung auf der Ausgabekarte (die Kartendimension), als auch die maximale Anzahl der SOM-Trainingsschritte t_{\max} bekannt gegeben werden. Viele Anwender arbeiten mit heuristischen Werten zwischen $t_{\max} = 4 \cdot |\mathbf{D}|$ und $t_{\max} = 7 \cdot |\mathbf{D}|$. Wegen der Möglichkeit zur Visualisierung der Nachbarschaftsbeziehungen der Eingabevektoren verwendet man in der Regel zwei-dimensionale Kohonenkarten. Unter Umständen werden die Eingabedaten so komprimiert, daß die Informationsdimension des Eingaberaumes die Kartendimension nicht übersteigt (s. hierzu [SRR94] und [Speck95]). Das Training der SOM wird gewöhnlich solange mit Kohonenkarten unterschiedlicher Neuronenanzahl (und evtl. unterschiedlicher Kartendimension) fortgesetzt, bis zufriedenstellende „Clusterergebnisse“ vorliegen. Die Detektierung der zugrundeliegenden Cluster erfolgt dabei durch den Standard SOM-Algorithmus nicht automatisch, sondern muß entweder durch eine erfahrene Person auf Grund des Betrachtens der visualisierten Nachbarschaftsbeziehungen erfolgen, oder dem Algorithmus werden cluster-erkennende Methoden nachgeschaltet, wie z. B. das an der Universität Tübingen entwickelte Verfahren „Clusot“ (s. [Bogd98], S. 99 ff.).

Im folgenden sollen für die vorliegenden bivariaten bzw. zwei-dimensionalen künstlichen Eingabedatensätze Ergebnisse der SOM präsentiert werden, die mit ein- und zwei-dimensionalen Kohonen-Karten mit jeweils unterschiedlichen Kartengrößen (Neuronenanzahl) realisiert werden können. Für den Eingabedatensatz `modes1`, der aus drei voneinander separierten Modi besteht, wird zunächst eine ein-

dimensionale 3x1 SOM benutzt, um eine optimale, aber rein theoretische Clustererkennung durch die SOM zu demonstrieren². Wie zu erwarten, adaptiert die SOM die Gewichte ihrer drei Neuronen im Verlauf des Trainings so, daß sie zu Trainingsende annähernd mit den Moduszentren der drei Eingabemodi zusammenfallen. t_{\max} wird mit $7 \cdot 300 \cong 2.000$ vorgegeben. Abb. 2.9. zeigt die finale Anordnung der 3 SOM-Neuronen, die über die festgelegte ein-dimensionale Kohonenkette angeordnet sind.

Sowohl EQE als auch ENT können während des Trainings minimiert werden: Der erwartete Quantisierungsfehler EQE nimmt bereits nach ca. 800 Trainingsschritten von ca. 0,3943 nach 0.0339 ab (s. Abb. B.3. (a)). Die Entropiewerte ENT beginnen bei 0, erreichen zu den Lernschritten 238, 239 und 243 den maximalen Wert 1,98 und kehren nach ca. 700 Lernschritten zum Optimum 0 zurück (s. Abb. B.3. (b)). Die Entropie ist zu Beginn des SOM-Trainings optimal, da alle SOM-Neuronen am Koordinatenursprung $(0,0)^T$ initialisiert werden und also alle Neuronen die gleiche Anzahl von Eingabeneuronen innerhalb ihrer Voronoi-Mengen umfassen. Das Training bricht nach den vorgegeben 2.000 Lernschritten bzw. nach ca. 5,48 sec ab.

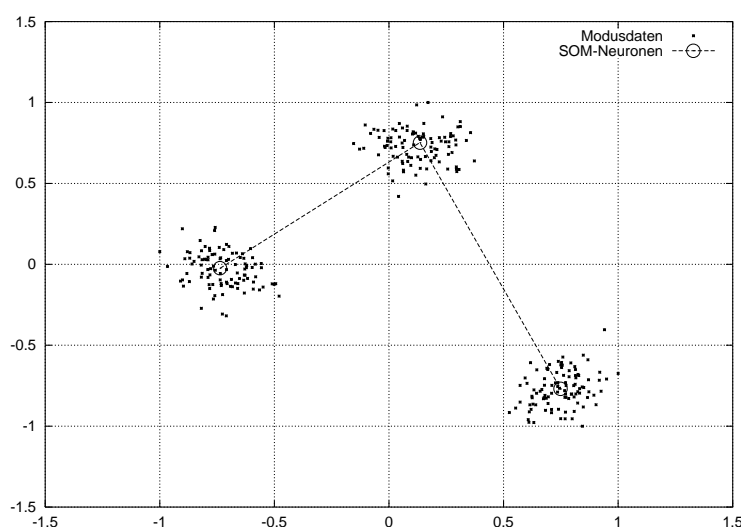


Abb. 2.9.: Optimales Ergebnis einer 3x1 SOM nach 2.000 Trainingsschritten für den Eingabedatensatz `modes1`.

Im weiteren soll die obige eher hypothetische SOM-Konfiguration durch eine realistischere ersetzt werden: Da bei Clusteranalyse-Verfahren vorausgesetzt wird, daß der Eingabedatensatz vollkommen unbekannt ist, bzw. daß weder Anzahl noch Form der Eingabemodi bekannt sind, wird die Ausgabekarten-Dimension zwei-dimensional gewählt und Versuche werden mit 25 (5x5) bzw. 100 (10x10) SOM-Neuronen durchgeführt. t_{\max} ist unabhängig von der Kartengröße und bleibt mit dem Wert 2.000 initialisiert. Das Training der 5x5 SOM bricht nach $\bar{\varnothing} 26,36 \pm 0,5$ sec, das Training der 10x10 SOM nach $\bar{\varnothing} 1$ min 37,24 sec $\pm 0,6$ sec ab.

Wie Abb. 2.10. (a), (b) und Abb. 2.11. (a), (b) zeigen, plaziert die SOM ihre Neuronen an Orte hoher Wahrscheinlichkeitsdichten, d. h. sie verteilt ihre Kartenneuronen so im Eingaberaum, daß diese möglichst gut die zugrundeliegenden Modi charakterisieren können. Bedingt durch den SOM-Algorithmus verbleiben jedoch 7 (28 %; Abb. 2.10.) bzw. 23 (23 %; Abb. 2.11.) Neuronen nach Abschluß des Trainings an Orten geringer Wahrscheinlichkeitsdichten (jeweils zu sehen im Zentrum des Eingaberaums zwischen den drei Eingabemodi)³. Diese Neuronen dienen nicht der Vektorquantisierung und ver-

2. Für alle Experimente wird die an der Universität Tübingen entwickelte Implementierung *kisom* benutzt.

3. Vergleichende Ergebnisse

schlechtern zudem die Entropiewerte ENT, da sie deutlich weniger Eingabevektoren innerhalb ihrer Voronoi-Mengen umfassen als dies Neuronen innerhalb der Modi tun können. Da im Verlauf des SOM-Trainings keine Neuronen verworfen werden, müssen die verbleibenden Neuronen möglichst gleichförmig an Orten hoher Wahrscheinlichkeitsdichten verteilt werden. Zwar repräsentieren sie dadurch die Form der zugrundeliegenden Modi recht gut, tragen jedoch, wie zuvor, nicht zur Vektorquantisierung bei: Eine echte Vektorquantisierung benötigt nur ein einziges Neuron, um ein gaußverteiltes Modus repräsentieren zu können.

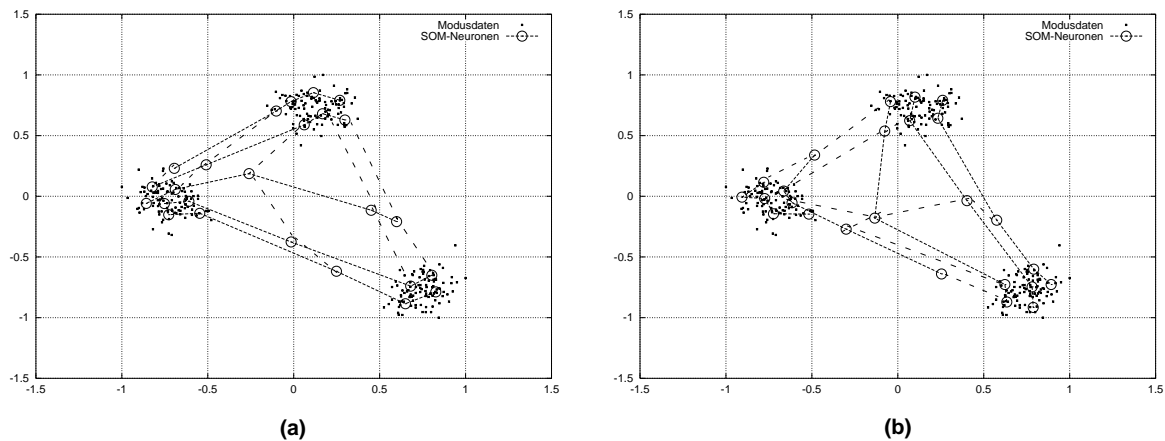


Abb. 2.10.: Beispielhafte Ergebnisse zweier 5x5 SOM nach 2.000 Trainingsschritten für den Eingabedatensatz `modes1`.

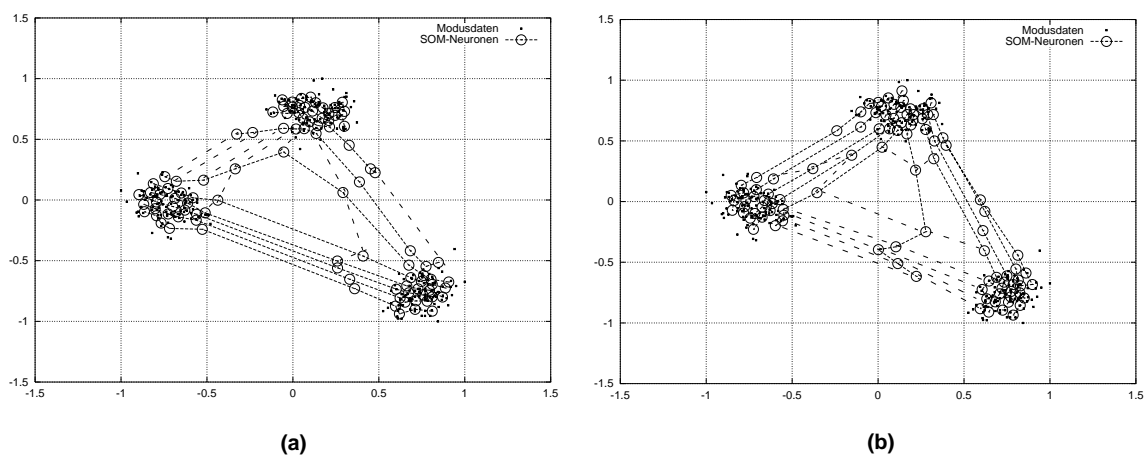


Abb. 2.11.: Beispielhafte Ergebnisse zweier 10x10 SOM nach 2.000 Trainingsschritten für den Eingabedatensatz `modes1`.

Der Quantisierungsfehler EQE kann trotz des schlechten vektorquantisierenden Verhaltens der 10x10 SOM im Verlauf des Trainings minimiert werden: Er wird von einem Startwert von $\varnothing 0,02 \pm 0,1$ nach 2.000 Lernschritten auf den Wert $\varnothing 0,005 \pm 0,004$ reduziert (s. a. Abb. B.4. (a)). Der Grund für die

3. Die gleichen Ergebnisse können mit Hilfe des „DemoGNG v1.3“ Java Applets von H. S. Loos und B. Fritze der Systems Biophysics, Institute for Neural Computation, Ruhr-Universität Bochum erhalten werden ([LoFr97], s. a. [HeRo98], S. 42 ff.).

schlechte Aussagekraft des EQE-Wertes für die Güte der Vektorquantisierung liegt darin begründet, daß Neuronen an Orten geringer Wahrscheinlichkeitsdichten immer den optimalen Wert 0 erreichen, da sie innerhalb ihrer Voronoi-Mengen keine Eingaben umfassen können und somit die Abstände der Eingabe zu diesen Neuronen immer 0 bleibt (vgl. hierzu auch Abschnitt 3., S. 16).

Aussagekräftiger erweist sich die Entropie, die mit Hilfe der ENT-Werte gemessen werden können: Die Entropie kann zwar maximiert werden bzw. ENT nimmt von einem Startwert $\bar{\varnothing} 150 \pm 10$ nach $\bar{\varnothing} 82 \pm 4$ ab, ENT bleibt jedoch auch nach Beenden des Trainings auf einem vergleichbar hohen Wert und erreicht nicht das Optimum 0. Zudem zeigen die Entropiewerte zum Abschluß der SOM-Trainingsphase recht hohe Varianzen (9 %; vgl. Abb. B.4. (b)), was darauf hindeuten mag, daß das SOM-Training im Bezug auf die Entropie nicht stabil genug ist.

Für den Eingabedatensatz `modes9`, der 1.800 2-dimensionale Vektoren enthält, soll die SOM nach $4 \cdot 1.800 \cong 7.000$ Schritten das Lernverfahren abbrechen. Der Karte werden $10 \times 10 = 100$ Neuronen vorgegeben. Ein beispielhaftes Ergebnis des Anlernens dieser SOM repräsentiert Abb. 2.12..

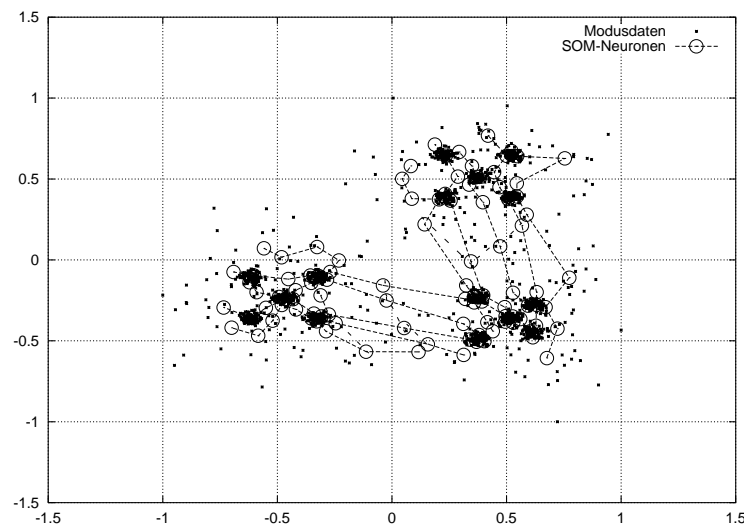


Abb. 2.12.: Beispielergebnis einer 10x10 SOM nach 7.000 Trainingsschritten für den Eingabedatensatz `modes9`.

Die Ergebnisse anhand des neuen Datensatzes `modes9` bestätigen die Ergebnisse mit Eingabe `modes1`: Die SOM plaziert für `modes9` die zur Verfügung stehenden Neuronen nicht nur an Orte hoher Wahrscheinlichkeitsdichten, also den drei globulären Eingabemodi, sondern auch an Orte geringer Wahrscheinlichkeitsdichten zwischen zwei globulären Modi oder am Rand eines globulären Modus' (vgl. Abb. 2.12.). Die meisten „Hits“ bekommen dabei solche SOM-Neuronen, die innerhalb der Varianz des Subclusters eines globulären Modus' zu liegen kommen: Diese SOM-Neuronen können am meisten Eingaben innerhalb ihrer Voronoi-Mengen umfassen bzw. relativ viele Eingaben werden auf dieses Neuron abgebildet. Die Trainingszeit beträgt $\bar{\varnothing} 33 \text{ min } 30 \text{ sec} \pm 48 \text{ sec}$.

Wiederum kann der erwartete Quantisierungsfehler EQE von einem primären Wert von $\bar{\varnothing} 0,01 \pm 0,005$ zu einem finalen Wert $\bar{\varnothing} 0,003 \pm 0,0004$ reduziert werden (vgl. Abb. B.5. (a)) und spiegelt damit die Tatsache wider, daß Neuronen an niedriger Wahrscheinlichkeitsdichte den Gesamtwert nicht verschlechtern können. Die Entropie fällt jedoch während des SOM-Trainings wesentlich schlechter aus und ENT wird zwar von $\bar{\varnothing} 125 \pm 10$ nach $\bar{\varnothing} 89 \pm 5$ reduziert, kann aber das Optimum 0 nicht annähernd erreichen. Zudem besitzt die Entropie zum Abschluß des Trainings deutliche Varianzen innerhalb verschiedener Trainingsläufe (s. Abb. B.5. (b)).

3. Vergleichende Ergebnisse

Wird die adaptierte Kohonenkarte aus Abb. 2.12. entfaltet, so daß benachbarte SOM-Neuronen benachbart in einem regelmäßigen Neuronengitter zu liegen kommen, so erhält man die alternative Darstellungsweise aus Abb. 2.13.: Sie zeigt zum einen die zwei-dimensionalen Neuronengewichte als Graphen, die euklidischen Distanzen zwischen benachbarten SOM-Neuronen als Balken unterschiedlicher Graustufen, sowie die Anzahl der Vektoren, die auf ein SOM-Neuron abgebildet werden, als „Hits“. Dabei wird ein Eingabevektor genau dann auf ein SOM-Neuron abgebildet und erzeugt einen „Hit“ für dieses Neuron, wenn die euklidische Distanz des Vektors zum gewinnenden Neurons kleiner ist als zu jedem anderen Neuron des Neuronengitters. Dies ist gleichbedeutend mit der Tatsache, daß der Eingabevektor in der Voronoi-Menge des gewinnenden SOM-Neurons liegt. Resultierende Cluster lassen sich aus dieser Graphik unter Berücksichtigung der Neuronendistanzen und Hits herauslesen.

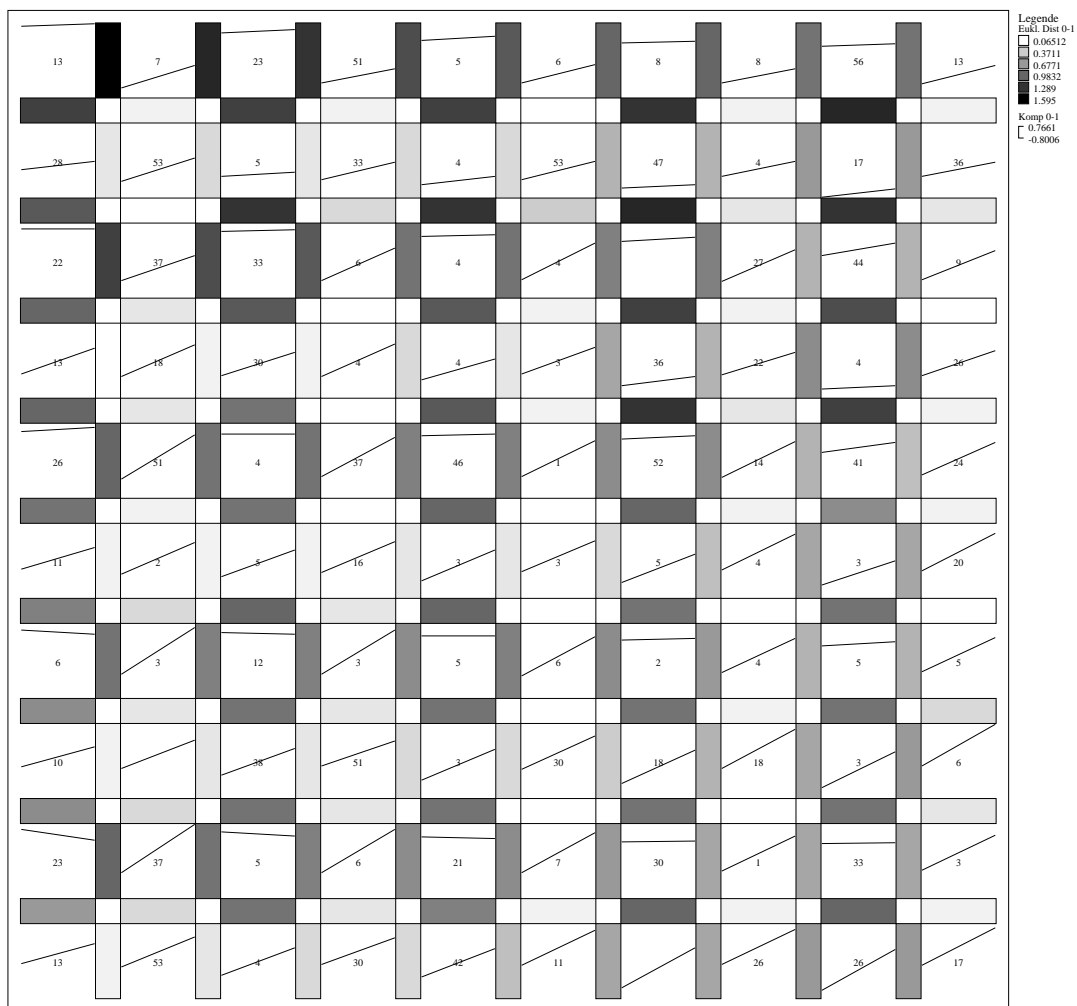


Abb. 2.13.: Alternative Darstellungsart der adaptierten 10x10 SOM aus Abb. 2.12. für Eingabedatensatz modes9.

Benutzt man eine 10x10 SOM, um den Datensatz modes12 zu repräsentieren, so ergibt sich nach $7 \cdot 700 \cong 5.000$ Lernschritten und ca. 9 min 44 sec \pm 2 sec Trainingszeit die Situation aus Abb. 2.14.

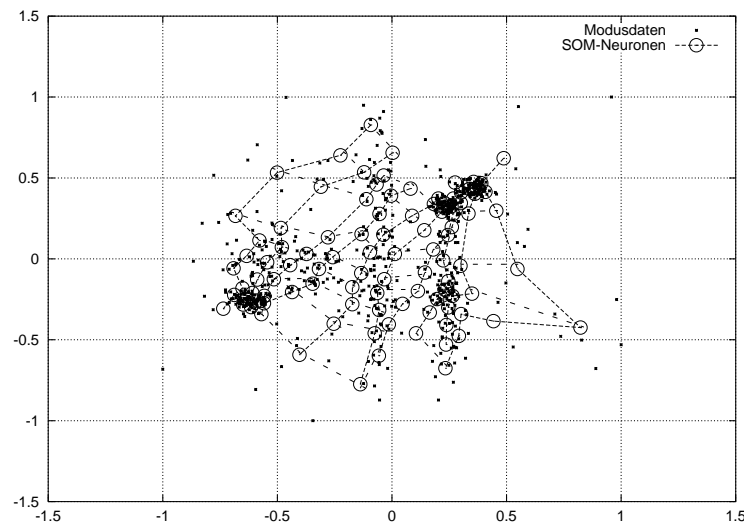


Abb. 2.14.: Beispielergebnis einer 10x10 SOM nach 5.000 Trainingsschritten für den Eingabedatensatz `modes12`.

`modes12` verdeutlicht am ehesten den Nachteil der SOM, Neuronen nicht nur an Orten hoher Wahrscheinlichkeitsdichten, sondern auch an Orten geringer Wahrscheinlichkeitsdichten zu platzieren. Im Beispielergebnis aus Abb. 2.14. liegen mindestens 13-15 SOM-Neuronen, d. h. 13-15 % aller Neuronen, außerhalb der normalen Varianzen der Eingabemodi (zu sehen am Rand des Eingaberaumes). Dieser Nachteil läßt sich vor allem an der Entropie messen, da SOM-Neuronen an Orten geringer Wahrscheinlichkeitsdichten wesentlich weniger Eingaben innerhalb ihrer Voronoi-Mengen umfassen können als Neuronen nahe der Modizentren.

Die gemessenen Werte der erwarteten Quantisierungsfehler EQE und die Entropiewerte ENT entsprechen im wesentlichen denen, die durch `modes9` gemessen werden können: Der erwartete Quantisierungsfehler EQE kann die Quantisierungsleistung der SOM nur bedingt wiedergeben und nimmt von $\varnothing 0,01 \pm 0,002$ nach 5.000 Trainingsschritten nach $\varnothing 0,0032 \pm 0,0004$ ab (s. Abb. B.6. (a)). Der Entropiewert ENT kann zum einen jedoch nicht minimiert werden und nimmt im gleichen Zeitraum nur von $\varnothing 105 \pm 20$ nach $\varnothing 74 \pm 6$ ab. Zum anderen zeigt die Entropie wiederum bei Beenden des SOM-Trainings große Varianzen (vg. Abb. B.6. (b)).

3.4. Top-Down HRCL

Das Top-Down HRCL bietet in einer ersten Hierarchiestufe einen „groben Überblick“ über den geclusterten Eingabedatensatz, d. h. detektiert entweder Cluster mit großen Varianzen oder faßt mehrere Cluster kleinerer Varianzen zu einem Cluster zusammen. Hierzu werden durch den Benutzer Umgebungsradien gewählt, die genügend groß sind. Die „grobe“ erste Ansicht des Datensatzes wird in den folgenden Top-Down Hierarchiestufen verfeinert: Jede weitere HRCL-Hierarchiestufe „zoom“ (durch geeignetes Skalieren) dabei in solche Bereiche des Datensatzes hinein, die durch die hyperkubischen Umgebungen der adaptierten HRCL-Neuronen definiert sind und versucht innerhalb des so neu definierten Eingabedatensatzes Cluster zu detektieren. Falls in den Eingabedatensätzen dieser hyperkubischen Umgebungen Cluster gefunden werden, so spricht man von Subclustern der „primären“ Superclustern. Die Vorgehensweise dieser hierarchischen Verfeinerung kann vor allem an den globulären oder

3. Vergleichende Ergebnisse

irregulären Datensätzen `modes9` bzw. `modes12` verdeutlicht werden (s. u.). Zunächst soll jedoch die generelle Arbeitsweise des Top-Down HRCL für eine hierarchische Stufe und dem einfachen multimodalen Datensatz `modes1` dokumentiert werden.

Das Top-Down HRCL adaptiert für den Eingabedatensatz `modes1` die 5 durch die Zellenclustering erhaltenen initialen Neuronen (vgl. Abb. 2.3.) so, daß je drei HRCL-Neuronen die Clustermittelpunkte der vorhandenen Cluster so gut wie möglich annähern und die restlichen zwei Neuronen verworfen werden (Neuronen-Pruning), da sie mehrere nacheinander folgende HRCL-Lernschritte unfixiert bleiben oder außerhalb des Bereichs $[-1,5;1,5]$ in einer der Dimensionen zu liegen kommen. Abb. 2.15. (a) zeigt beispielhaft die Bewegung der adaptierten Neuronen („Neuronenspur“; die verworfenen Neuronen sind nicht dargestellt), Abb. 2.15. (b) zeigt die finale Position der drei Neuronen mit ihren Radien nach Beenden des HRCL-Trainings. Die Quantisierungsfehler beginnen jeweils bei einem Wert von 0,0324 und

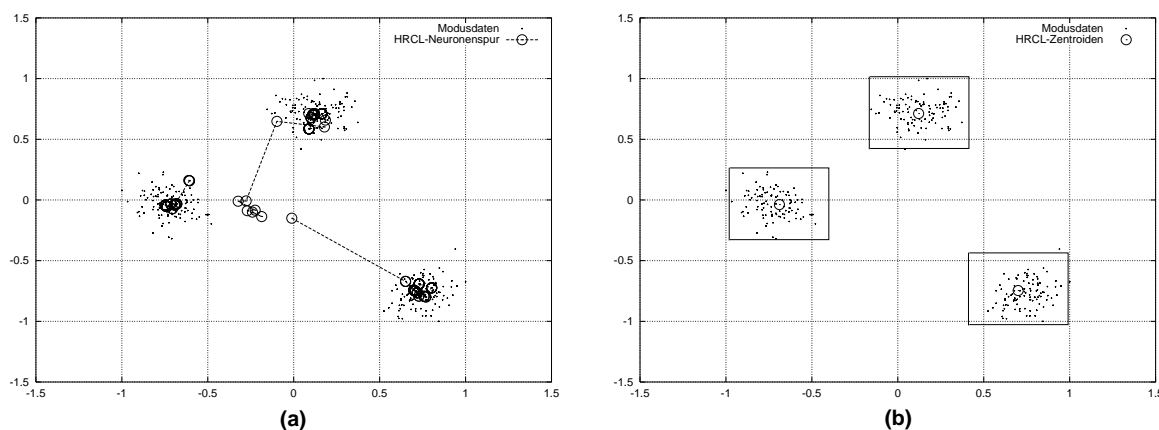


Abb. 2.15.: (a) Neuronenspuren der 3 adaptierten Neuronen und (b) Ergebnis der 0. Hierarchiestufe des Top-Down HRCL mit $r = 0,3$ für `modes1`

enden nach dem Trainings bei $0,025 \pm 0,001$ (s. Abb. B.7. (a)). Die Entropiewerte ENT starten bei 2,1 und erreichen schon nach 20-30 HRCL-Trainingsschritten den optimalen Wert 0 (s. Abb. B.7. (b)). Das Top-Down HRCL erkennt für `modes1` 3 Hierarchiestufen und benötigt für die Adaption der Neuronen zur 0. Hierarchiestufe ca. 69 ± 30 Trainingsschritte in einer Trainingszeit von ca. $7,6 \pm 2$ sec (Alle Werte sind über 10 Trainingsläufe gemittelt.).

Die Fähigkeit des Top-Down HRCL, eine hierarchische Verfeinerung des Eingabedatensatzes vorzunehmen, soll insbesondere an den Datensätzen `modes9` und `modes12` demonstriert werden. Für `modes9` und Eingaberadius $r = 0,25$ werden insgesamt 3 Hierarchiestufen entdeckt. In der 0. Hierarchiestufe werden durch die Zellenclustering 8 initiale HRCL-Neuronen generiert, die nach ca. 634 ± 240 HRCL-Trainingszyklen 3 (globuläre) Cluster entdecken, für welche 3 HRCL-Neuronen mit Radius $r = 0,25$ und Überlappung von 0 % an deren Mittelwerte platziert werden. (s. Abb. 2.16.). Die restlichen 5 Neuronen werden im Verlauf des Trainings verworfen. Die Subcluster der detektierten globulären Cluster aus Hierarchiestufe 0 werden jeweils in der nächsten Hierarchiestufe erkannt. Hierzu wird in jedes der detektierten globulären Cluster aus Hierarchiestufe 0 „hineingezoomt“, indem die Eingabedaten des entsprechenden Clusters, die durch den Radius $r = 0,25$ und Umgebung $U_{c_i}(r)$ festgelegt sind, auf das reelle Intervall $[-1;1]$ skaliert werden. Die jeweils 5-6 initialen HRCL-Neuronen der 1. Hierarchiestufe werden nach ca. 70-80 HRCL-Lernschritten auf die Subclusterzentroiden platziert (s. Abb. 2.17. (a)-(c)). In der 2. Hierarchiestufe werden insgesamt 6 Subsubcluster detektiert, für welche HRCL-Neuronen an deren Mittelwerte platziert werden (hier nicht gezeigt). Die Trainingszeit für alle HRCL-Hierarchiestufen beträgt ca. 12 ± 5 min.

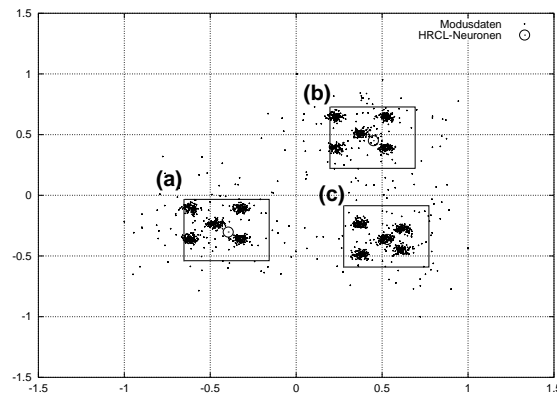


Abb. 2.16.: Ergebnis der 0. Hierarchiestufe des Top-Down HRCL mit $r = 0,25$ für `modes9`

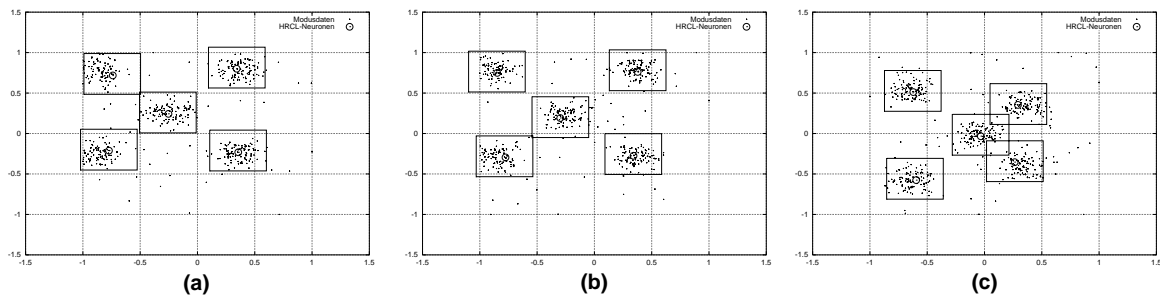


Abb. 2.17.: Ergebnisse der 1. Hierarchiestufe des Top-Down HRCL (Verfeinerung der Ergebnisse aus Abb. 2.16.) mit $r = 0,25$ für `modes9`

Der Quantisierungsfehler für `modes9` kann in der 0. Hierarchiestufe des HRCL Top-Down Verfahrens noch minimiert werden, obwohl 5 Neuronen während des Trainings verworfen werden. Er nimmt während der ca. 634 HRCL-Trainingszyklen nur gering von 0.0465 auf ca. $0.0462 \pm 0,0008$ ab und kann sich am Ende des Trainingsverlaufs stabilisieren (s. Abb. B.8.(a)). Für die Entropie gelten günstigere Bedingungen, da sie nicht von der Anzahl der Neuronen abhängig ist. Sie kann für `modes9` und die 0. Hierarchiestufe von einem Anfangswert von 2.5866 schon nach ca. 190 Lernschritten (ENT2) auf das Optimum 0 minimiert werden (s. Abb. B.8.(b)).

Für den Eingabedatensatz `modes12` und vorgegebenem Radius $r = 0,3$ erkennt das HRCL Top-Down, ausgehend von 3 initialen HRCL-Neuronen, in der 0. von insgesamt 3 Hierarchiestufen 3 globale Cluster. Cluster (a) aus Abb. 2.18. wird in der nachfolgenden Hierarchiestufe 1 in 4 Subcluster zerlegt (s. Abb. 2.19. (a)), ebenso kann das Cluster (b) in 6 Subcluster und das globuläre Cluster (c) in 2 Subcluster unterteilt werden (s. Abb. 2.19.(b) und (c)). Besonders dichte Subcluster der 1. Hierarchiestufe, d. h. solche kleiner Kovarianzen, können weiter in Subsubcluster aufgespalten werden: Subcluster (i) aus Abb. 2.19.(a) wird im nächsten hierarchischen Schritt in 6 Subsubcluster zerlegt, Subcluster (ii) und (iii) aus Abb. 2.19.(c) werden jeweils in 4 Subsubcluster zerlegt (s. Abb. 2.20.(i)-(iii)). Der Quantisierungsfehler nimmt von 0.1072 über ca. 273 ± 77 Lernschritte der 0. Top-Down Hierarchie nach ca. 0.0727 ab (s. a. Abb. B.9.(a)). Gleichzeitig kann über denselben Zeitraum die Entropie erhöht werden, d. h. ENT wird von 1 nach ca. 0.3725 verringert (s. a. Abb. B.9.(b)). Die gesamte Lerndauer beträgt über alle Hierarchiestufen ca. 1 min 55 sec.

3. Vergleichende Ergebnisse

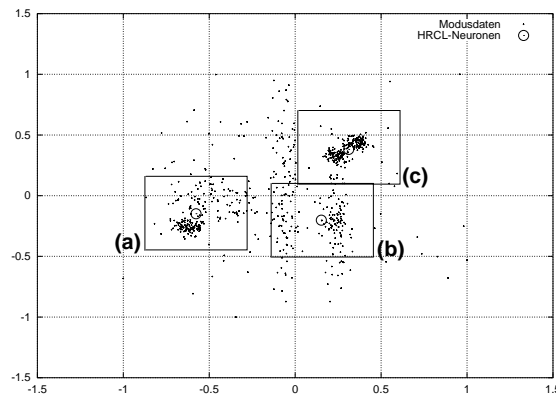


Abb. 2.18.: Ergebnis der 0. Hierarchiestufe des Top-Down HRCL mit $r = 0,3$ für `modes12`

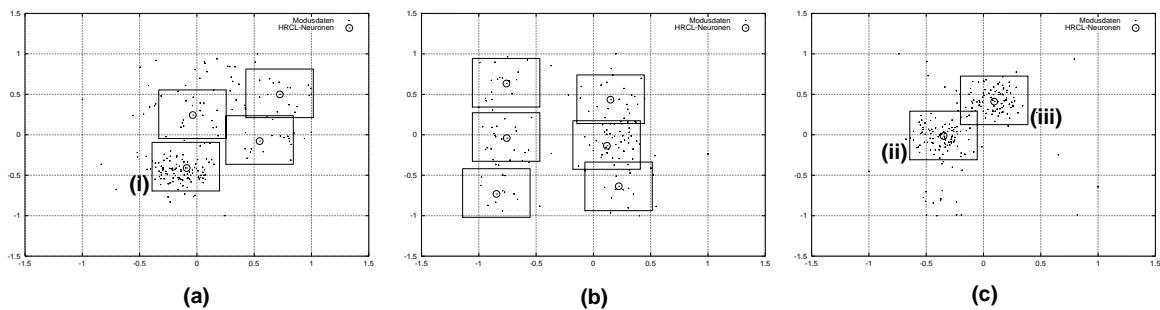


Abb. 2.19.: Ergebnis der 1. Hierarchiestufe des Top-Down HRCL (Verfeinerung der Ergebnisse aus Abb. 2.21.) mit $r = 0,3$ für `modes12`

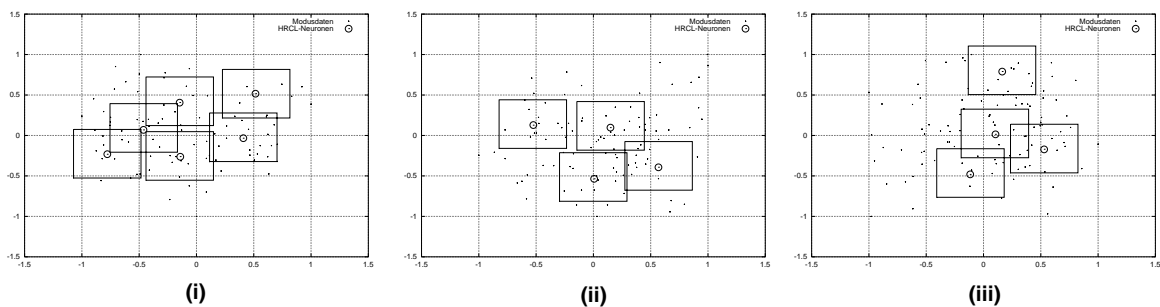


Abb. 2.20.: Ergebnis der 2. Hierarchiestufe des Top-Down HRCL (Verfeinerung der Ergebnisse aus Abb. 2.19.(i)-(iii)) mit $r = 0,3$ für `modes12`

Das HRCL Top-Down erkennt für denselben Datensatz mit einem Radius von $r = 0,15$ in der 0. Hierarchie 5 Cluster, von denen 2 ((a) und (b) in Abb. 2.21.) als globuläre Cluster in der 1. Hierarchiestufe weiter verfeinert werden können. Für Cluster (a) werden 2 Subcluster (s. Abb. 2.22. (a)), für das globuläre Cluster (b) aus Abb. 2.21. werden 7 Subcluster mit Radien $r = 0,15$ erkannt (s. Abb. 2.22.(a) und (b)). Der Quantisierungsfehler nimmt in der 0. Hierarchiestufe trotz dreier verworfener HRCL-Neuronen von 0.0563 nach ca. 781 Lernschritten nach 0.0524 ab. Gleichzeitig kann die Entropie zunehmen,

bzw. nimmt ENT von 2.2286 nach 2.0429 ab. Die Hierarchisierung bricht nach der 1. Hierarchiestufe ab. Es werden also keine weiteren Subsubcluster detektiert und die gesamte HRCL-Trainingszeit beträgt ca. 9 min 32 sec.

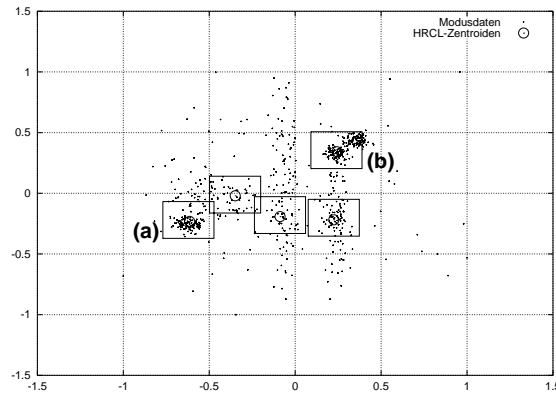


Abb. 2.21.: Ergebnis der 0. Hierarchiestufe des Top-Down HRCL mit $r = 0,15$ für `modes12`

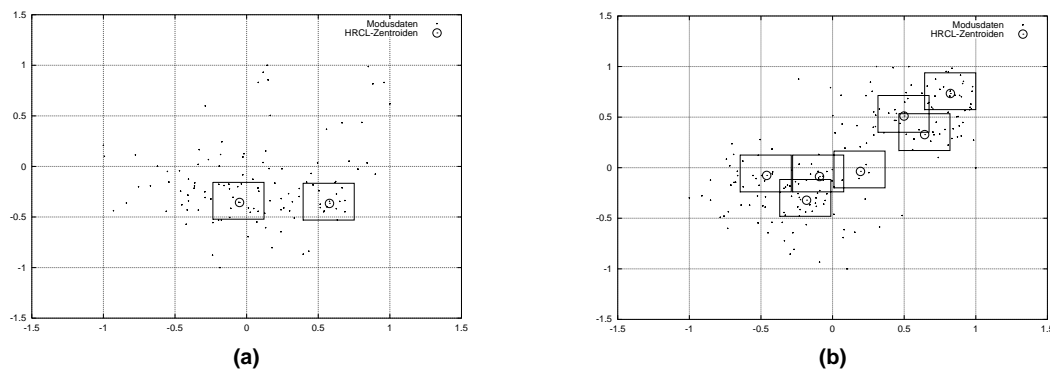


Abb. 2.22.: Ergebnis der 1. Hierarchiestufe des Top-Down HRCL (Verfeinerung der Ergebnisse aus Abb. 2.21.) mit $r = 0,15$ für `modes12`

3.5. Bottom-Up HRCL

Im Gegensatz zum Top-Down HRCL werden im Bottom-Up Ansatz in der ersten Hierarchiestufe kleinste Details oder Wahrscheinlichkeitsdichten erkannt. D. h. es werden die tatsächlichen Verteilungen der zugrundeliegenden Cluster wiedergegeben. Diese Detailansichten bzw. HRCL-Neuronen werden in weiteren Bottom-Up Hierarchiestufen zu einem einzigen Cluster zusammengefaßt, falls dies sinnvoll und möglich ist. Besteht der Datensatz andererseits aus einer Reihe globulärer Cluster, d. h. aus Clustern, die wiederum aus Subclustern bestehen, so werden zu allen weiteren Bottom-Up Hierarchiestufen Cluster der bereits bearbeiteten Hierarchiestufen sinnvoll zu sogenannten Supercluster zusammengefaßt. Die Möglichkeit der Bottom-Up Hierarchisierung, Supercluster aus Clustern zu „formen“, kann vor allem an dem multi-modalen globulären Datensatz `modes9` demonstriert werden. Die Möglichkeit, aus der Repräsentation unterschiedlicher Clusterverteilungen auf das zugrundeliegende Cluster zu schließen, wird insbesondere durch den Eingabedatensatz `modes12` verdeutlicht (s. u.).

3. Vergleichende Ergebnisse

Im Bottom-Up Ansatz werden die initialen Neuronen nicht durch die Zellenclustering gebildet, sondern es werden $N := \sqrt{|\mathbf{D}|}$ HRCL-Neuronen zufällig auf Eingaben ξ aus \mathbf{D} plaziert (s. Abschnitt 2.2.). Für den Eingabedatensatz `modes1`, der aus $3 \cdot 100$ Eingabevektoren besteht, werden demzufolge 17 initiale Neuronen generiert. Eine exemplarische Verteilung dieser initialen Neuronen zeigt Abb. 2.23.(a) (Die Neuronen-Umgebungen $U_c(r)$ sind der besseren Übersichtlichkeit halber nicht dargestellt.). Sie dienen dem HRCL Bottom-Up als Seeds und werden in den nachfolgenden HRCL-Lernschritten so angepaßt, daß sie mit den Moduszentren der vorhandenen Cluster zusammenfallen. Dabei werden überschüssige Neuronen verworfen. Abb. 2.23.(b) zeigt eine finale Verteilung von 3 HRCL-Neuronen der 0. HRCL Bottom-Up Hierarchiestufe und Radius $r = 0,3$, wie sie i. d. R. nach ca. 53 ± 16 HRCL-Lernschritten auftritt. Das Bottom-Up HRCL erkennt für obigen Eingabedatensatz und Radius nur eine Hierarchiestufe und bricht also nach der 0. Hierarchiestufe bereits mit der Berechnung ab. Die Rechenzeit beträgt ca. 25 ± 14 sec.

Die Tatsache, daß im obigen Beispiel 82 % der initialen Neuronen verworfen werden, erschwert das Auswerten des Quantisierungsfehlers erheblich. Der Quantisierungsfehler erhöht sich für den Datensatz `modes1` und Radius $r = 0,3$ von ca. $0,0112 \pm 0,003$ auf den Wert $0,0672 \pm 0,15$ (s. a. Abb. B.10.(a)). Gleichzeitig wird die Entropie, die unabhängig von der Anzahl der Neuronen ist, bereits nach 34 bis 69 Lernschritten maximiert bzw. ENT verschwindet (s. Abb. B.10.(b)). Wird jedoch der Quantisierungsfehler EQE durch den normierten Quantisierungsfehler NQE aus Gl. (1.14) ersetzt, so wird nicht nur durch Abb. 2.23.(b) deutlich, daß das HRCL für `modes1` eine optimale Vektorquantisierung erreichen kann (s. Abb. B.11.): Der normierte Quantisierungsfehler NQE fällt von $6,345 \cdot 10^{-4} \pm 1,5 \cdot 10^{-4}$ nach $4,055 \cdot 10^{-4} \pm 1,5 \cdot 10^{-4}$ ab.

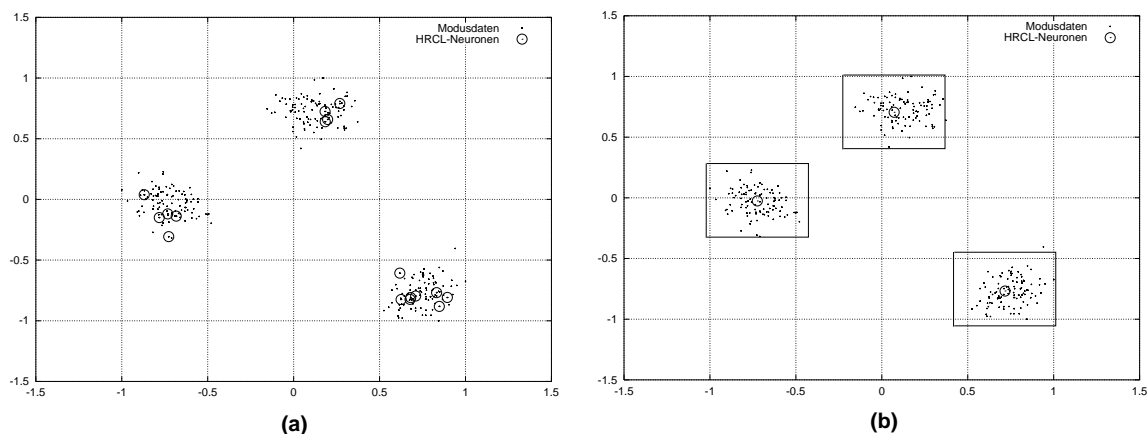
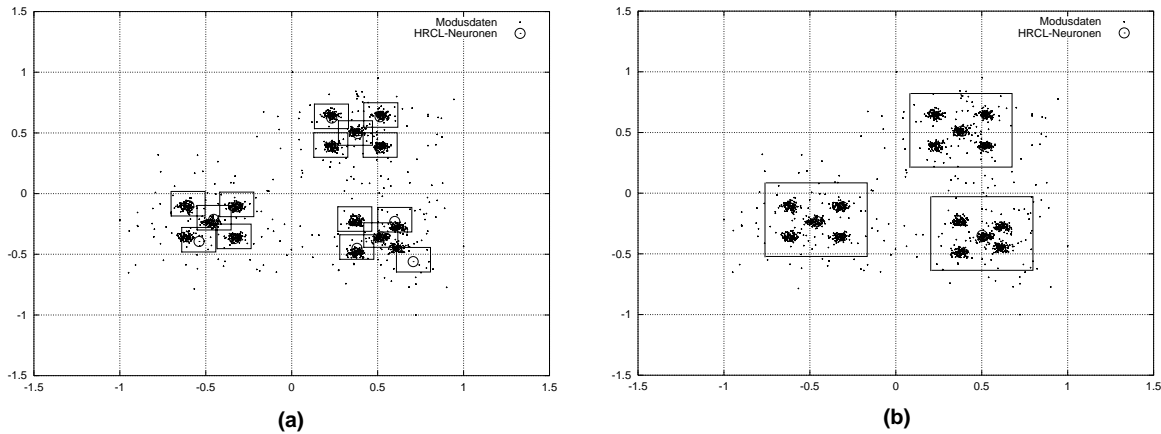


Abb. 2.23.: (a) Initiale Position der 17 HRCL-Neuronen und (b) Ergebnis der 0. Hierarchiestufe des Bottom-Up HRCL mit $r = 0,3$ für `modes1`

Für den globulären Eingabedatensatz `modes9`, initialem Radius $r = 0,1$ und einer Erhöhung des Radius r um $\lambda_{\text{radius}} = 3$ in jeder weiteren Hierarchiestufe, erkennt das Bottom-Up HRCL zwei Hierarchiestufen. Ein Beispielergebnis der beiden detektierten Hierarchiestufen spiegelt Abb. 2.24.(a) und (b) wider. In der 0. Hierarchiestufe werden 42 initiale HRCL-Neuronen auf ca. 15-20 Neuronen verkleinert und gleichzeitig auf Orte hoher Wahrscheinlichkeitsdichten plaziert, wodurch ebensoviele Cluster detektiert werden. In der 1. Bottom-Up Hierarchiestufe werden ca. $\text{int}(\sqrt{15}) = 3$ bis $\text{int}(\sqrt{20}) = 4$ initiale HRCL-Neuronen auf 3 Neuronen mit Radius $r = \lambda_{\text{radius}} \cdot 0,1 = 0,3$ verkleinert. Die resultierenden 3 Neuronen stellen Supercluster der 15-20 detektierten Cluster aus Hierarchiestufe 0 dar bzw. die Cluster der 0. Hierarchiestufe sind Subcluster der 3 Cluster aus Hierarchiestufe 1.

Der Quantisierungsfehler nimmt in der 1. Hierarchiestufe nach ca. 202 ± 108 HRCL-Lernschritten von

ca. $0,165 \pm 0,19$ nach $0,041 \pm 0,04$ ab (s. Abb. B.12.(a)). Gleichzeitig kann für dieselbe hierarchische Ebene die Entropie vergrößert bzw. ENT verkleinert werden: ENT nimmt von ca. $0,752 \pm 0,84$ nach $0,0079 \pm 0,02$ ab (s. Abb. B.12.(b)). Die gesamte Laufzeit für beide Hierarchiestufen beträgt ca. 14 ± 1 min.



**Abb. 2.24.: Ergebnis der (a) 0. und (b) 1. Hierarchiestufe des Bottom-Up HRCL mit initialem Radius $r = 0,1$ und $\lambda_{\text{radius}} = 3$ für `modes9`:
Detektion von 15 Clustern und 3 Superclustern.**

Für den irregulären multi-modalen Datensatz `modes12`, initialem Radius $r = 0,1$ und einer Radiusverdoppelung durch $\lambda_{\text{radius}} = 2$ in jeder weiteren Hierarchiestufe, erkennt das Bottom-Up HRCL gewöhnlich 3 Hierarchiestufen. Zur 0. Hierarchiestufe werden $\text{int}(\sqrt{7} \cdot 100) = 26$ initiale HRCL-Neuronen auf ca. 17 ± 2 Neuronen bzw. detektierte Cluster reduziert. Das Bottom-Up HRCL benötigt hierzu ca. 175 ± 75 Lernschritte. In der 1. hierarchischen Stufe werden ca. $\text{int}(\sqrt{17}) = 4$ initiale Neuronen zu den Orten hoher Wahrscheinlichkeitsdichten hin adaptiert. Die folgende 3. Hierarchiestufe adaptiert schließlich $\sqrt{4} = 2$ HRCL-Neuronen. Die Trainingszeit für alle Hierarchiestufen beträgt ca. 9 min 30 sec ± 1 min. Die HRCL-Neuronen der 0. Hierarchiestufe werden so im Eingabedatenraum verteilt, daß sie die zugrundeliegenden Cluster möglichst exakt in ihrer räumlichen Verteilung widerspiegeln können. Abb. 2.25.(a) zeigt beispielhaft diese clusterbeschreibende Fähigkeit des Bottom-Up HRCL. Nicht nur die 3 normalverteilten „dichten“ Cluster an den Positionen $(-15, -8)^t$, $(8, 9)^t$ und $(11, 12)^t$ mit Varianzen $(1, 1)^t$ (s. Abschnitt 3.1.) werden erkannt, sondern auch alle weiteren Cluster werden durch bis zu 6 HRCL-Neuronen beschrieben. Die clusterbeschreibenden Ergebnisse der 0. Hierarchiestufe werden in der nachfolgenden 1. Hierarchiestufe sinnvoll zusammengefaßt. Die 1. Hierarchiestufe bildet also Supercluster von Clustern der 0. Hierarchiestufe, wobei die HRCL-Neuronen der 1. Hierarchiestufe wiederum an Orte hoher Wahrscheinlichkeitsdichten bewegt werden: Die Ergebnisse aus Abb. 2.25.(b) verdeutlichen, daß die beiden Cluster an den Positionen $(8, 9)^t$ und $(11, 12)^t$ zu einem Supercluster zusammengefaßt werden. Zudem versucht die 1. Hierarchiestufe das Cluster an der Stelle $(-15, -8)^t$ und Varianz $(1, 1)^t$ mit dem Cluster an $(-11, -2)^t$ und deutlich größerer Varianz $(4, 3)^t$ sinnvoll zusammenzufassen. Schließlich entdeckt ein HRCL-Neuron der 1. Hierarchiestufe die Zentren der beiden Cluster mit Mittelwert $(8, -8)^t$ und Varianz $(1, 6)^t$, sowie das Cluster mit Mittelwert $(0, 0)^t$ und mit in y-Richtung verlängerter Varianz $(1, 12)^t$ (s. Abschnitt 3.1.). Die 2. Hierarchiestufe faßt die Cluster aus der 1. hierarchischen Stufe zu 2 Superclustern zusammen und positioniert die HRCL-Neuronen an Orte hoher Wahrscheinlichkeitsdichten, d. h. vor allem nahe des Zentrums des „dichten“ Clusters mit Zentrum $(-15, -8)^t$ bzw. zwischen Clusterzentroid $(8, 9)^t$ und $(11, 12)^t$ (s. Abb. 2.26.).

3. Vergleichende Ergebnisse

Der Quantisierungsfehler EQE kann während des Trainings der HRCL-Neuronen der 0. Hierarchiestufe nicht verkleinert werden, da ca. 35 % der initialen Neuronen verworfen werden: Er steigt von $2,418 \cdot 10^{-2} \pm 0,2 \cdot 10^{-2}$ nach $2,558 \cdot 10^{-2} \pm 0,4 \cdot 10^{-2}$ an (s. Abb. B.13.(a)). Der normierte Quantisierungsfehler NQE wird jedoch während des gleichen Zeitraums von ca. $7,408 \cdot 10^{-4} \pm 1,0 \cdot 10^{-4}$ nach $5,837 \cdot 10^{-4} \pm 1,1 \cdot 10^{-4}$ verkleinert (s. Abb. B.14.). Ebenso kann die Entropie vergrößert bzw. ENT verkleinert werden: ENT nimmt von $14,8 \pm 1$ nach $11,04 \pm 1,4$ ab (s. Abb. B.13.(b)).

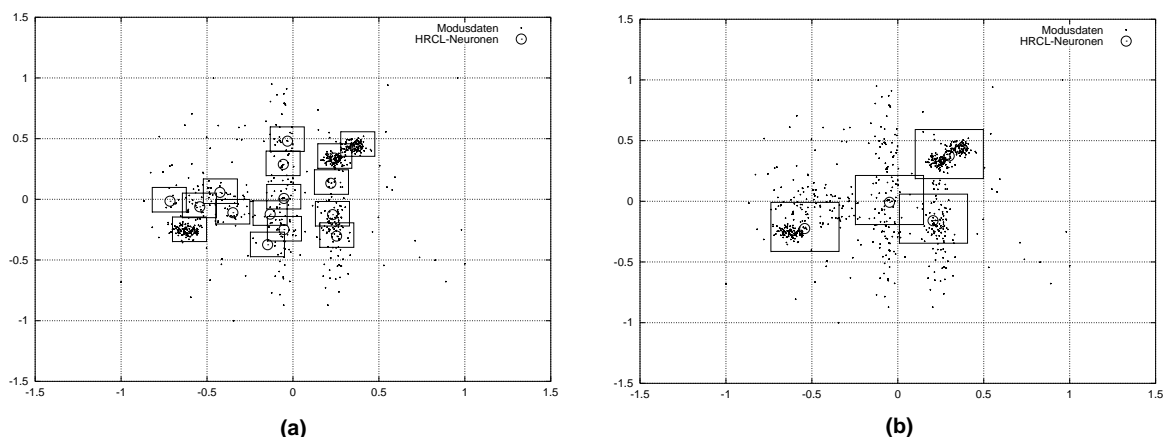


Abb. 2.25.: Ergebnis der (a) 0. und (b) 1. Hierarchiestufe des Bottom-Up HRCL mit initialem Radius $r = 0,1$ und $\lambda_{\text{radius}} = 2$ für modes12: Detektion von 16 bzw. 4 Clustern.

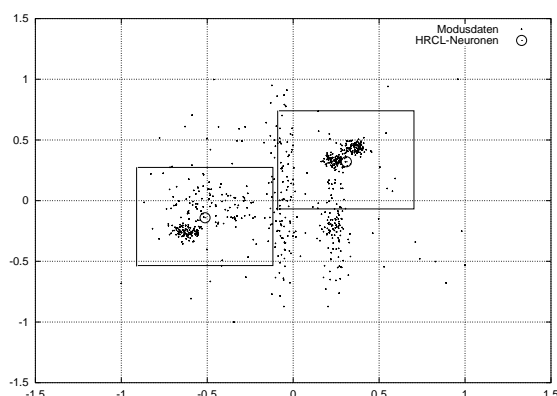


Abb. 2.26.: Ergebnis der 2. Hierarchiestufe des Bottom-Up HRCL mit initialem Radius $r = 0,1$ und $\lambda_{\text{radius}} = 2$ für modes12: Detektion von 2 Clustern.

Die Anzahl der hierarchischen Verfeinerungstufen (d. h. die Tiefe der HRCL-Hierarchisierung) kann für modes12 vergrößert werden, indem der initiale Radius auf $r = 0,05$ halbiert, λ_{radius} entsprechend mit dem Wert $\lambda_{\text{radius}} = 1,5$ belegt und zudem $N := |\mathbf{D}|^{2/3}$ statt $N := \sqrt{|\mathbf{D}|}$ initiale HRCL-Neuronen als auch $N^{2/3}$ statt \sqrt{N} HRCL-Neuronen in jeder weiteren Hierarchiestufe generiert werden (s. Abschnitt 2.2.). Entsprechend werden 4 statt 3 Hierarchiestufen erzeugt und zur 0. Hierarchiestufe 78 auf ca. 50-46 HRCL-Neuronen, in der 1. Stufe ca. 13 auf 12 Neuronen reduziert und in der 2. Stufe 5 Neuronen, in der 3. Stufe 2 Neuronen und evtl. in einer 4. Hierarchiestufe 1 Neuron adaptiert. Das Training zur 0. Hierarchiestufe beträgt ca. 68 HRCL-Lernschritte, die Gesamtlerndauer für alle 4 Hierarchiestufen beträgt ca. 21 min 10 sec.

Abb. 2.27.(a), (b) und Abb. 2.28.(a), (b) zeigen beispielhafte Ergebnisse aller 4 detektierten Hierarchiestufen. Es bleibt festzuhalten, daß zu allen Hierarchiestufen die Auflösung des zugrundeliegenden Datensatzes größer ist, d. h. die 0. Hierarchiestufe erkennt noch besser als zuvor kleinste Details bzw. kann noch besser die Clusterverteilungen repräsentieren (Abb. 2.27.(a) und (b) im Vergleich zu Abb. 2.25.(a)). Das gleiche gilt für alle weiteren Hierarchiestufen, nämlich für die Ergebnisse aus Abb. 2.28.(a) im Vergleich zu Abb. 2.25.(b) bzw. Ergebnis aus Abb. 2.28.(b) in Relation zu Abb. 2.26.

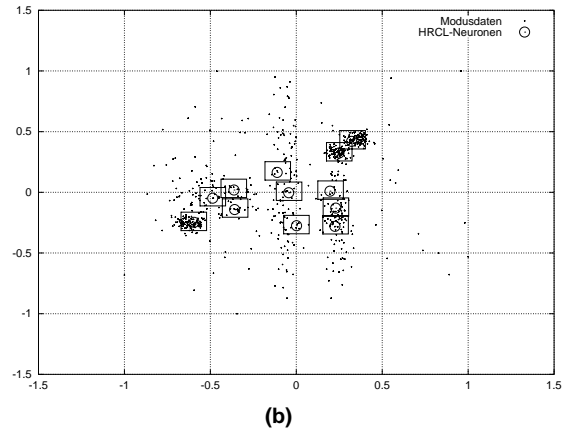
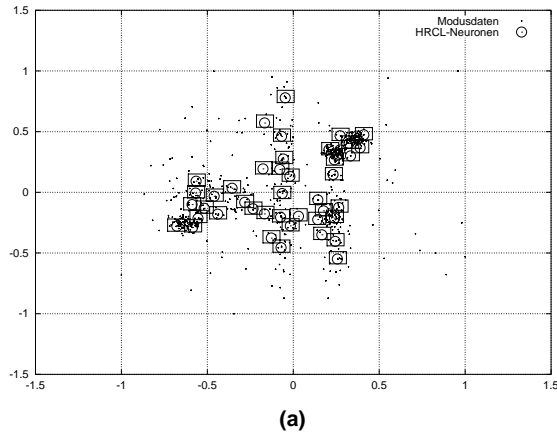


Abb. 2.27.: Ergebnis der (a) 0. und (b) 1. Hierarchiestufe des Bottom-Up HRCL mit initialem Radius $r = 0,05$ und $\lambda_{\text{radius}} = 1,5$ für modes_{12} : Detektion von 46 bzw. 12 Clustern.

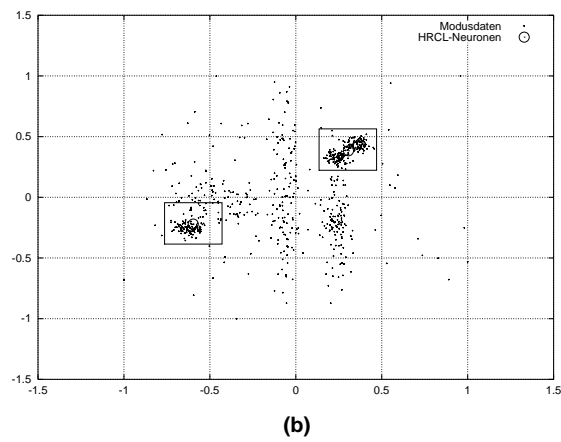
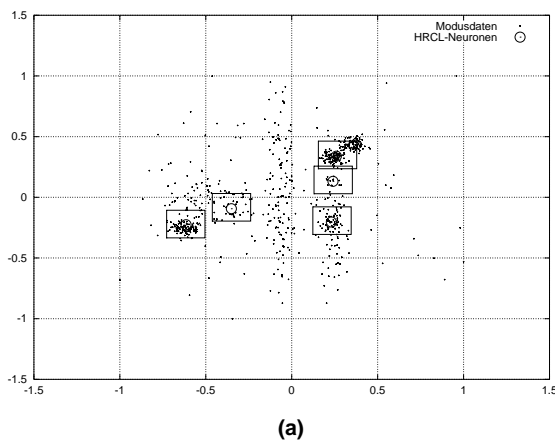


Abb. 2.28.: Ergebnis der (a) 2. und (b) 3. Hierarchiestufe des Bottom-Up HRCL mit initialem Radius $r = 0,05$ und $\lambda_{\text{radius}} = 1,5$ für modes_{12} : Detektion von 5 bzw. 2 Clustern.

4. Diskussion der Ergebnisse

Clusteranalyse-Verfahren versuchen Cluster oder Modi, d. h. Orte hoher Wahrscheinlichkeitsdichten des Eingabektorraums, zu finden. Dabei ist in vielen realen Anwendungen weder bekannt, ob, und wenn ja, wieviele Modi der Datensatz enthält, noch besitzt das Clusterverfahren ein Wissen über die Form oder Verteilung der zugrundeliegenden Cluster. In all diesen Fällen muß das Clusterverfahren selbständig sowohl die *Anzahl* bzw. *Mittelwerte* als auch die *Verteilung* der Eingabemodi ermitteln können. In fast allen Fällen kann man davon ausgehen, daß Cluster nicht notwendigerweise normal-verteilt vorliegen. Der Beschreibung der Clusterverteilung kommt also zunächst eine wichtige Aufgabe zu. Während viele Verfahren die Clusterverteilung gut repräsentieren können, vernachlässigen sie es jedoch, die Anzahl der Cluster exakt wiederzugeben. Man sagt auch, daß eine echte Vektorquantisierung der Maximierung der Entropie entgegenschläuft (vgl. a. [HeRo98], S. 30 ff. und [Fritz97]). Dieser Effekt ist sowohl bei dem untersuchten statistischen Verfahren, dem Single-Pass SMART-Retrieval, als auch beim unüberwacht lernenden oder sich selbst organisierenden neuronalen Netz, der Selbst-Organisierenden Karte (SOM), zu beobachten und macht sich insbesondere bei nicht normal-verteilter oder „irregulär“ geformten Eingabemodi, wie z. B. bei `modes12` (s. Abb. 2.7. (b)), bemerkbar. Die SOM darf dabei im engeren Sinne nicht als Clusterverfahren bezeichnet werden, da sie sich vorrangig zur Aufgabe genommen hat, Nachbarschaftsbeziehungen des Eingabektorraums nachbarschaftserhaltend auf eine Ausgabekarte abzubilden (s. [Koh82, Koh84] bzw. [HeRo98], S. 33 ff.). Aus Gründen der Visualisierung der erhaltenen Nachbarschaftsbeziehungen werden oft zwei-dimensionale Ausgabekarten verwendet, auch wenn diese Kartendimension nicht immer optimal in Bezug zur Eingabe ist (nämlich dann, wenn die Informationsdimension des Eingaberaums größer als zwei ist; vgl. [SRR94, Speck95]). Die SOM wird schließlich oft dazu benutzt, Cluster aus der selbst-organisierten Karte herauszulesen.

Die Clusterergebnisse, die anhand der multi-modalen künstlichen Eingaben erzielt werden konnten und für das SMART-Retrieval in Abschnitt 3.2., für die SOM in Abschnitt 3.3. gezeigt wurden, belegen offensichtlich obige Aussage: Sowohl das SMART-Retrieval, als auch die SOM⁴ plazieren eine Anzahl von Cluster-Prototypen bzw. Neuronen an Orte hoher Wahrscheinlichkeitsdichten, können jedoch normal-verteilte Cluster nicht durch nur ein einziges Neuron (Cluster-Prototyp) repräsentieren (vgl. Abb. 2.8. sowie Abb. 2.10. und Abb. 2.11. für `modes1`). Da also keine echte Vektorquantisierung stattfindet (für die SOM u. a. aus dem Grund, weil sie sich eher auf eine topologisch korrekte Abbildung der Eingabevektoren konzentriert), muß die Anzahl der zugrundeliegenden Cluster in einem weiteren Schritt aus dem „Clusterergebnis“ herausgelesen werden, bzw. dies müssen geeignete Werkzeuge (wie z. B. das in Abschnitt 3.3. erwähnte „Clusot“) nachträglich vornehmen. Darüberhinaus ist das statistische SMART-Retrieval, wie schon von den Autoren des Verfahrens in [SaMc83], S. 235 (s. a. [HeRo98], S. 15) erwähnt, stark von der Reihenfolge der seriell zu verarbeitenden Eingabevektoren abhängig und plaziert auch Cluster-Prototypen an außenliegende Modusdaten, sogenannte „Ausreißer“. Ersteres macht sich vor allem dadurch bemerkbar, daß der erwartete Quantisierungsfehler EQE sich auch nach Beenden der Adaption nicht stabilisieren kann (s. Abb. B.1.), letzteres beeinflusst deutlich die Entropie, die nicht maximiert werden kann (d. h. ENT kann nicht minimiert werden; s. Abb. B.2.).

Die SOM wiederum plaziert zwar Neuronen an Orte hoher Wahrscheinlichkeitsdichten, jedoch auch an Stellen mit nur sehr geringen Dichten, z. B. zwischen allen Eingabemodi (s. Abb. 2.10. und Abb. 2.11.

4. Dies bezieht sich auf eine SOM, die mehr Neuronen besitzt als Modi im Eingabedatensatz vorhanden sind und tritt also in vielen realen Fällen auf, in denen die Anzahl (und Verteilung) der Eingabemodi unbekannt ist.

für `modes1`). Dies macht sich zwar nicht im erwarteten Quantisierungsfehler bemerkbar, da SOM-Neuronen an geringen Wahrscheinlichkeitsdichten nur unwesentlich zum erwarteten Quantisierungsfehler beitragen, jedoch verbleibt der Entropiewert ENT nach Beenden des SOM-Trainings auf einem relativ hohen Wert und variiert deutlich von Trainingslauf zu Trainingslauf (s. Abb. B.4. (b)). Ein weiterer Nachteil des SOM-Verfahrens ist es, daß zum einen die Anzahl der SOM-Neuronen dem Verfahren im voraus bekannt gegeben werden müssen, und zum anderen SOM-Neuronen während oder nach dem Training nicht verworfen werden können. Dadurch ist der SOM-Algorithmus gezwungen, die zur Verfügung stehenden Neuronen möglichst gleichverteilt über alle Modi zu verteilen, auch wenn eine geringere Anzahl von Neuronen zur Beschreibung der Anzahl und Verteilung der Modi genügen würde.

Setzt man nun vereinfachend voraus, daß die Eingabedatensätze grundsätzlich aus normal-verteilten Modi gleicher Kovarianzen bestehen, so lassen sich durchaus die Anzahl der vorhandenen Cluster bestimmen und also eine echte Vektorquantisierung vornehmen. Dies wird im Ansatz des hier vorgestellten Hierarchischen Radius-basierten Competitive Learning (HRCL) versucht. Dabei approximiert die n -dimensionale hyperkubische Umgebung um jedes HRCL-Neuron, abhängig vom durch den Benutzer vorzugebenden Radius, die Kovarianz, zugleich nähert das HRCL-Neuronengewicht die Erwartungs- oder Mittelwerte der zugrundeliegenden Modi an. Da nicht davon ausgegangen werden kann, daß alle Modi des (skalierten) Eingaberaumes den approximierten Kovarianzen des HRCL-Verfahrens genügen, werden die nur grob erkannten Modi im Top-Down HRCL weiter verfeinert, um deren Details zu erkennen. Details sind entweder tatsächlich im Eingabedatensatz existierende Subcluster vorhandener „primärer“ Cluster (wie sie in `modes9` vorliegen; vgl. Abb. 2.7. (a)), oder aber Repräsentationen der Verteilung existierender (Sub-)Cluster (vgl. `modes12`, Abb. 2.7. (b)). Dieser HRCL-Ansatz, der eine Hierarchie von detektierten Clustern, Subclustern, Subsubclustern bzw. Clusterverteilungen generiert, kann als eine Art „Multiresolutions-Clusterung“ angesehen werden: Nach der ersten HRCL-Hierarchiestufe, die eine nur grobe geclusterte Ansicht des Eingabedatensatzes liefert, werden die nur grob erkannten Cluster in weiteren Hierarchiestufen zu ihren Details „aufgelöst“, bis schließlich in der letzten möglichen HRCL-Hierarchiestufe (Sub-)Clusterverteilungen beschrieben werden können (s. Abb. 2.16. und Abb. 2.17. bzw. Abb. 2.18., Abb. 2.19. und Abb. 2.20.).

Der HRCL-Algorithmus bricht dabei automatisch die hierarchische Auflösung, aufgrund der Modalität der (zur jeweiligen Hierarchie gehörigen) Eingabe, ab. Auf jeder HRCL-Hierarchiestufe bleibt der erwartete Quantisierungsfehler niedrig, und ENT kann schon nach wenigen HRCL-Trainingsschritten minimiert werden (s. Abb. B.7., Abb. B.8. und Abb. B.9.). Im Gegensatz zur Selbst-Organisierenden Karte ermittelt das HRCL die Anzahl der initialen Neuronen (Seeds) mit Hilfe einer modifizierten Zellenclustering selbständig und verwirft automatisch auf jeder Hierarchiestufe solche Neuronen, die nicht zur Vektorquantisierung beitragen. Zudem wird das HRCL-Training bereits zu einem Zeitpunkt abgebrochen, wenn sich der erwartete Quantisierungsfehler nicht mehr merklich ändert, d. h. wenn vermutet werden kann, daß ein (lokales) Fehlerminimum gefunden wurde. Vergleicht man in diesem Zusammenhang die Trainingszeiten des SMART-Retrievals, der SOM (in der Implementierung des „kison“) und des HRCL (bis zum Beenden der 0. Hierarchiestufe) für denselben Eingabedatensatz `modes1`, so wird deutlich, daß aufgrund der a-priori festzulegenden Anzahl von Trainingsschritten, das SMART-Retrieval und die SOM in etwa die gleichen Trainingszeiten benötigen, nämlich 1 min 25 sec (SMART; vgl. Abschnitt 3.2.) bzw. 1 min 37 sec (10x10 SOM; vgl. Abschnitt 3.3.), das (Top-Down) HRCL jedoch nach ca. 7,6 sec bereits das Training der 0. Hierarchiestufe beenden kann (vgl. Abschnitt 3.4.). Die Trainingszeiten der SOM sind dabei quadratisch von der zuvor festzulegenden Anzahl an Neuronen abhängig, und die 5x5 SOM benötigt für denselben Datensatz `modes1` lediglich ca. 26 sec. Die Trainingszeiten des HRCL wachsen mit der Anzahl an automatisch generierten Hierarchiestufen. Für den multimodalen globulären Datensatz `modes9` ergeben sich für die 10x10 SOM und das HRCL folgende Trainingszeiten: 33 min 30 sec (bei 7.000 SOM-Lernschritten) bzw. 12 min (für alle 3 HRCL-Hierarchiestufen). Vergleiche der Laufzeitverhalten sollten jedoch immer unter Berücksichtigung der

4. Diskussion der Ergebnisse

eingesetzten Implementierung betrachtet werden: In der aktuellen Arbeit wurde die relativ langsame SOM-Implementierung „kisom“, sowie ein Byte-Code erzeugendes (und also ebenso langsames) Perl-Skript für das SMART-Retrieval und das HRCL benutzt. Die Performanz aller drei Implementierungen lassen sich sicherlich optimieren.

Cluster(mittelwerte) müssen schließlich im HRCL-Verfahren nicht zusätzlich aus den Ergebnissen errechnet werden, sondern stehen sofort dem Benutzer zur Verfügung.

Ist das Auflösungsvermögen der ersten HRCL-Hierarchiestufe, aufgrund des eingegebenen Radius' oder Eingabedatensatzes, ungenügend, so können auch in weiteren HRCL-Auflösungsstufen die Ergebnisse der ersten Stufe nicht verbessert werden. Für diesen Fall wurde die Top-Down Variante des HRCL durch das Bottom-Up HRCL ersetzt: Das Bottom-Up HRCL versucht, im Gegensatz zur Top-Down Variante, kleinste Wahrscheinlichkeitsdichten des Eingabedatensatzes zu detektieren, d. h. die Verteilung oder aber (Sub-)Subcluster der zugrundeliegenden Cluster widerzuspiegeln. Solcherart detektierte „primäre Cluster“ (oder besser: die Repräsentationen der Clusterverteilungen) werden in einer folgenden HRCL-Hierarchiestufe sinnvoll zusammengefaßt, so daß jeder Clusterprototyp der folgenden Hierarchiestufe den Erwartungswert eines Modus' repräsentiert, dessen Verteilung oder Subcluster bereits durch die primären Cluster beschrieben wurde. Solche Clusterprototypen werden auch als „Supercluster“ bezeichnet, die genannte Art der Hierarchisierung auch als „Abstraktion“ (vgl. Abb. 2.23. und insbesondere Abb. 2.24. bis Abb. 2.28.).

Das Bottom-Up HRCL ist, im Gegensatz zum Top-Down HRCL, nicht so sehr abhängig von der richtigen Wahl des Radius' und kann eventuelle Probleme des Skalierens der Eingabe zu jeder neuen Hierarchiestufe vermeiden, da das Bottom-Up HRCL immer auf demselben Eingabedatensatz operiert. Aus dem gleichen Grund hat jedoch das Bottom-Up HRCL ein schlechteres Laufzeitverhalten als die Top-Down Variante und läßt sich zudem auf prozeduraler Ebene nicht parallelisieren. Insgesamt kann jedoch gesagt werden, daß das Bottom-Up HRCL die Verteilung der zugrundeliegenden Modi noch besser wieder spiegeln kann als das Top-Down HRCL.

Für das Bottom-Up HRCL wird besonders deutlich, daß der erwartete Quantisierungsfehler EQE nicht in der Lage ist, die tatsächliche Güte der erreichten Vektorquantisierung zu messen. Eine optimale Vektorquantisierung ist bereits dann erreicht, wenn für jedes (globuläre) Cluster ein einziges Neuron in dessen Zentrum platziert wird. Der Quantisierungsfehler ist jedoch abhängig von der Anzahl der generierten Neuronen und nimmt mit dieser ab. Dies bedeutet einerseits, daß der Quantisierungsfehler verfälscht wird, je mehr Neuronen generiert werden, andererseits wird der Quantisierungsfehler verschlechtert, wenn während des Trainings Neuronen verworfen werden (s. a. Abschnitt 3., S. 14 ff.). Aus diesem Grund wurde insbesondere für das Bottom-Up HRCL, das bis zu 82 % der initialen Neuronen verwirft (s. Ergebnisse für `modes1` in Abschnitt 3.5.) ein normierter Quantisierungsfehler NQE eingeführt, in dessen Berechnung die Anzahl der existierenden HRCL-Neuronen mit eingeht (s. Gl. (1.14) und Gl. (1.15)). Der normierte Quantisierungsfehler NQE ermöglichte es, die visualisierten Vektorquantisierungsergebnisse durch ein objektives Maß zu quantifizieren (s. Abb. B.11. und Abb. B.14.).

Das Hierarchische Radius-basierte Competitive Learning ist, trotz des Namens, keine *hierarchische Clusterung* im Sinne der Kategorisierung nach Lance und Williams [LaWi67] (s. a. [JaDu88], S. 55 ff. und [HeRo98], S. 20), sondern läßt sich als neuronales Competitive Learning Verfahren der *exklusiven, intrinsischen, partitionierenden Klassifikation (Clusterung)* zuordnen: Das Clusterergebnis jeder „HRCL-Hierarchie- oder Auflösungsstufe“ dient als Eingabe zu einer partitionierenden Klassifikation der nächsten HRCL-Hierarchiestufe. Als Eingabe zur Clusterung jeder HRCL-Auflösungsstufe dient eine $m \times n$ - Matrix, bestehend aus m Eingabe- oder Dokumentenvektoren der Dimension n . Das HRCL kombiniert dabei die Vorteile der partitionierenden Klassifikation, die vor allem bei großen Eingabemengen ihre Stärke zeigt, mit den hierarchischen Auflösungseigenschaften der hierarchischen Clusterung, die es dem Benutzer ermöglicht, rasch das Clusterergebnis auszuwerten.

Abschließend sollen beispielhafte Ergebnisse weiterer Competitive Learning Verfahren anhand des multimodalen Datensatzes aus [HeRo98], Abb. 1.10. (a), S. 43 gezeigt werden, um sie mit den zuvor behandelten Verfahren zu vergleichen. Alle Competitive Learning Verfahren sind Implementierungen des Byte-Code erzeugenden „DemoGNG v1.3“ Java Applets von H. S. Loos und B. Fritzsche der Systems Biophysics, Institute for Neural Computation, Ruhr-Universität Bochum [LoFr97]. Es sollen lediglich die visualisierbaren Clusterergebnisse diskutiert werden, ohne auf die Auswirkungen auf den (normierten) erwarteten Quantisierungsfehler EQE bzw. NQE, die Entropie (ENT) oder die benötigte Trainingszeit näher einzugehen.⁵

Das Neuronale Gas (NG; s. [HeRo98], Abschnitt 6.2.5., S. 41) mit den Eingabeparametern $\lambda_i = 10$, $\lambda_f = 1$, $\varepsilon_i = 0,3$, $\varepsilon_f = 0,05$, $t_{\max} = 40.000$ und 20 Eingabeneuronen platziert, wie die SOM, ihre 20 NG-Neuronen sowohl an dichte als auch an weniger dichte Orte des Eingabevektorraums, wobei, im Vergleich zur SOM, die zugrundeliegenden Modiverteilungen deutlich besser wiedergegeben werden können. Neuronen können jedoch nicht verworfen werden, was unter anderem zur Folge hat, daß normalverteilte Modi nicht durch nur ein einziges NG-Neuron charakterisiert werden können. Es findet also keine echte Vektorquantisierung statt, und das Verfahren benötigt für ein akzeptables Ergebnis eine inakzeptable Rechenzeit von ca. einer Stunde (vgl. Abb. 2.29. (a) und (b)).

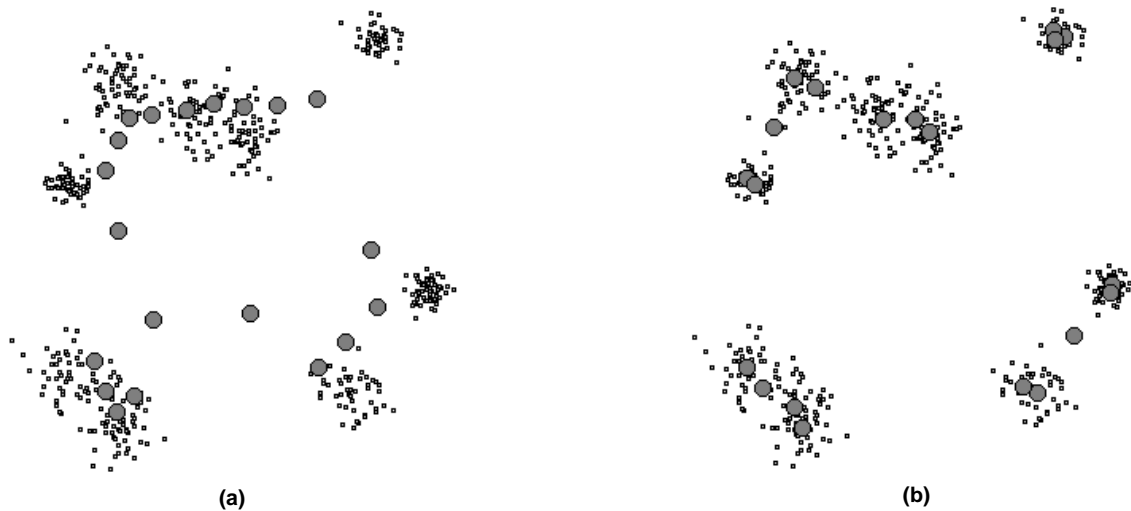


Abb. 2.29.: Ergebnisse für das Neuronale Gas (NG) mit 20 Eingabeneuronen nach (a) ca. 30 min und (b) ca. 1 h Trainingszeit.

Für das Growing Grid (GG; vgl. [HeRo98], Abschnitt 6.2.2.) mit 50 Eingabeneuronen und den Parametern $\lambda_g = 30$, $\lambda_f = 100$, $\varepsilon_i = 0,1$, $\varepsilon_f = 0,005$ und $\sigma = 0,9$ können die Ergebnisse aus Abb. 2.30. (a) und (b) erzielt werden. Das finale Ergebnis, dargestellt in Abb. 2.30. (b) kann mit denen der SOM verglichen werden: Auch das GG konzentriert sich eher auf eine topologisch korrekte Abbildung des (möglicherweise hochdimensionalen) Eingabevektorraums auf eine zwei-dimensionale Ausgabekartenstruktur. Dabei werden GG-Neuronen nicht nur an Orte hoher, sondern auch an Orten niedriger Wahrscheinlichkeitsdichten (z. B. zwischen allen Eingabemodi) platziert. Das GG benötigt nur etwa 20 % der NG-Trainingszeiten.

5. Die angegebenen Trainingszeiten sind besonders stark von der graphischen Visualisierung der Ergebnisse abhängig und sollen darum lediglich als Vergleichswerte innerhalb der DemoGNG-Umgebung verstanden werden.

4. Diskussion der Ergebnisse

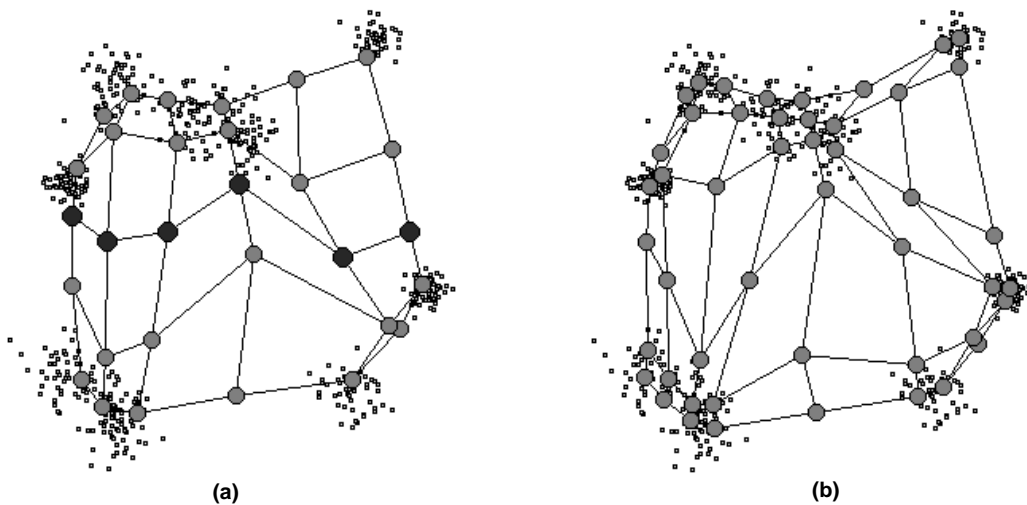


Abb. 2.30.: Ergebnisse des Growing Grids (GG) mit 50 Eingabeneuronen nach (a) ca. 3 min und (b) ca. 20 min Trainingszeit (inkl. 100 % Fine-Tuning).

Das GNG (s. [HeRo98], Abschnitt 6.2.3.), dessen Ergebnisse in Abb. 2.31. (a) - (c) dargestellt sind, kann sowohl eine echte Vektorquantisierung des Eingabedatensatzes, als auch eine sinnvolle Repräsentation der Modiverteilung vornehmen: Das GNG mit maximaler Neuronenzahl 50 und Parametern $\lambda = 600$, $\text{age_max}_{(s_1, s_2)} = 88$, $\epsilon_b = 0,05$, $\epsilon_n = 6,0 \text{ E-}4$, $\alpha = 0,5$ und $\beta = 5,0 \text{ E-}4$ platziert nach ca. 20 min (s. Abb. 2.31. (b)) ungefähr so viele Neuronen in den Eingaberaum wie normal-verteilte Modi vorhanden sind. Die Neuronengewichte sind nach 20 min Trainingszeit so adaptiert, daß sie ungefähr mit den Modizentren zusammenfallen. Dies entspricht einer optimalen Vektorquantisierung. Wird das GNG-Training fortgesetzt und nach ca. 50 min erneut unterbrochen, so repräsentieren die GNG-Neuronen bzw. deren Gewichte die Verteilungen der zugrundeliegenden Modi (s. Abb. 2.31. (c)). Beide Aufgaben können jedoch nicht zugleich erzielt werden, sondern sind abhängig vom Zeitpunkt des GNG-Trainingsabbruchs: Wird das GNG-Training frühzeitig abgebrochen, so repräsentieren die GNG-Neuronen Modizentren, wird das GNG-Training später abgebrochen, so repräsentieren die Neuronen die Modiverteilungen.

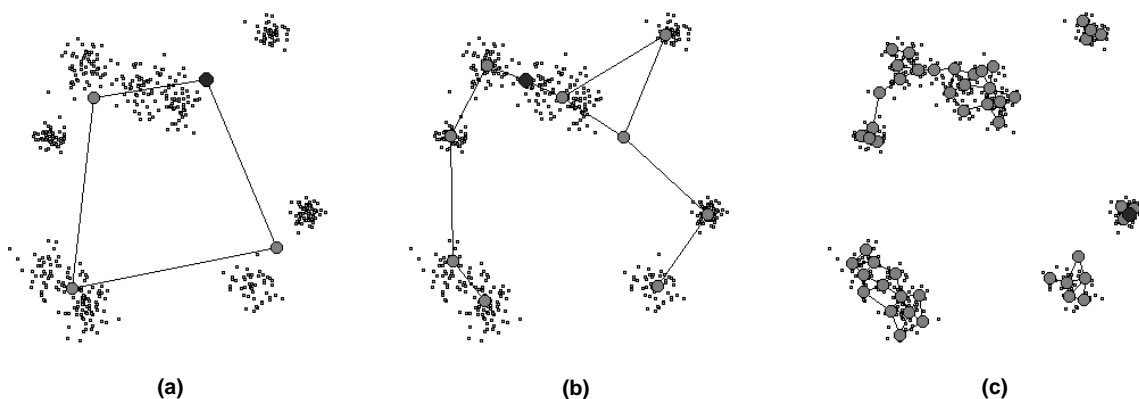


Abb. 2.31.: Ergebnisse des Growing Neural Gas (GNG) mit 50 Eingabeneuronen nach (a) ca. 2 min, (b) ca. 20 min und (c) ca. 50 min Trainingszeit.

5. Zusammenfassung

Diese Arbeit stellt das Hierarchische Radius-basierte Competitive Learning (HRCL) in zwei Varianten, dem Top-Down HRCL sowie dem Bottom-Up HRCL, vor. Das HRCL detektiert als neuronales Clusteranalyse-Verfahren die einem Eingabedatensatz zugrundeliegenden Modi oder Cluster, d. h. Orte hoher Wahrscheinlichkeitsdichten. Die Clusteranalyse wird automatisch durch das HRCL in weiteren Hierarchiestufen verfeinert: Im Top-Down HRCL werden somit weitere Details der (nur grob) detektierten „primären Cluster“ erkannt, bis schließlich in einer letzten Hierarchiestufe die Cluster-Verteilungen repräsentiert werden können. Das Bottom-Up HRCL hingegen versucht kleinste Wahrscheinlichkeitsdichten einer ersten HRCL-Hierarchiestufe in weiteren hierarchischen Schritten sinnvoll zu sogenannten Superclustern zusammenzufassen.

Beide Varianten des HRCL zeichnen sich dadurch aus, daß den Verfahren keine initiale Anzahl von HRCL-Neuronen bekannt gegeben werden muß. Zudem verwirft das HRCL automatisch solche Neuronen, die nicht zur Vektorquantisierung beitragen und bricht das HRCL-Training jeder Hierarchiestufe schon zu einem Zeitpunkt ab, wenn sich der erwartete Quantisierungsfehler nicht mehr merklich ändert, d. h. wenn erwartet werden kann, daß das Verfahren ein (lokales) Fehlerminimum gefunden hat.

Die Vorteile des HRCL zum untersuchten statistischen Single-Pass Clusteranalyse-Verfahren, dem SMART-Retrieval, lassen sich stichpunktartig folgendermaßen wiedergeben:

- Keine Abhängigkeit von der Reihenfolge der einzugebenden Eingabevektoren
- Echte Vektorquantisierung: Das SMART-Retrieval kann lediglich die Verteilung der Cluster wiedergeben, nicht die Erwartungs- oder Mittelwerte der Cluster.

Die vektorquantisierenden Vorteile des HRCL gegenüber der Selbst-Organisierenden Karte sind:

- Echte Vektorquantisierung: Die SOM bietet keine echte Vektorquantisierung
- Direkte Ausgabe der Clustermittelwerte zu jeder Auflösungsstufe
- Der Eingabedatensatz muß nicht notwendigerweise eine Informationsdimension von maximal 2 haben (bei Verwendung zwei-dimensionaler SOM-Ausgabekarten).
- Nachteil gegenüber der SOM: keine Visualisierung der Nachbarschaftsbeziehungen zwischen Eingabevektoren möglich

Zusätzlich bietet das HRCL folgende Eigenschaften:

- „Multiresolutions-Clusterung“: Automatische hierarchische Ausgabe der verschiedenen „Auflösungen“ des Eingabedatensatzes: Globuläre Cluster können von Clusterzentren über Subclusterzentren bis hin zu Subclusterverteilungen aufgelöst werden.
- Automatischer Abbruch sowohl des hierarchischen Aufbaus als auch des HRCL-Trainings zu jeder hierarchischen Stufe
- Automatische Initialisierung und Pruning aller HRCL-Neuronen

6. Ausblick

Da das Hierarchische Radius-basierte Competitive Learning dazu benutzt werden soll, eine Datensammlung zu clustern, die aus einigen tausend HTML-Seiten bestehen mag, muß das HRCL Cluster auch in Eingaberäumen detektieren können, deren Eingabedimensionen sehr viel höher als zwei ist. HTML-Dokumente werden nämlich über Termindizierverfahren kodiert, die gewöhnlich bis zu 10.000-dimensionale Dokumentenvektoren (auch „Dokumenten-Profile“ oder „profiles“ genannt) erzeugen. Vor allem aus Gründen der Effizienz werden diese sehr hoch-dimensionalen Eingabevektorräume mit Hilfe geeigneter Komprimierverfahren auf Dimensionen von ca. 10-100 verkürzt.

Um die Einsetzbarkeit des HRCL bei höherdimensionalen Räumen an einem Beispiel zu verdeutlichen, werden an dieser Stelle Ergebnisse gezeigt, die mit einem künstlich-erzeugten 10-dimensionalen multimodalen Eingabedatensatz erzielt werden kann: Der Eingabedatensatz `modes13` besteht dabei, als 10-dimensionale Erweiterung von `modes1` (vgl. Abb. 2.2. und Abb. 2.3.), aus drei deutlich voneinander separierten 10-dimensionalen Eingabemodi. `Modes13` besitzt folgende in Abschnitt 3.1. beschriebene Parameter:

$m = 3$, $n = 2$, $M' = 100$, sowie folgende drei Modusmittelwerte und -varianzen: $((1, 7, 1, 7, 1, 7, 1, 7, 1, 7)^t, (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^t)$, $((-7, 0, -7, 0, -7, 0, -7, 0, -7, 0)^t, (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^t)$ und $((7, -7, 7, -7, 7, -7, 7, -7, 7, -7)^t, (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^t)$. Projiziert man jeweils die Dimensionen $(d, d + 1)$, d ungerade und $(d, d + 2)$, d ungerade orthogonal auf ein zwei-dimensionales Koordinatensystem, so erhält man die in Abb. 2.32. (a) und Abb. 2.32. (b) wiedergegebenen Ausgaben für den Eingabedatensatz `modes13`.

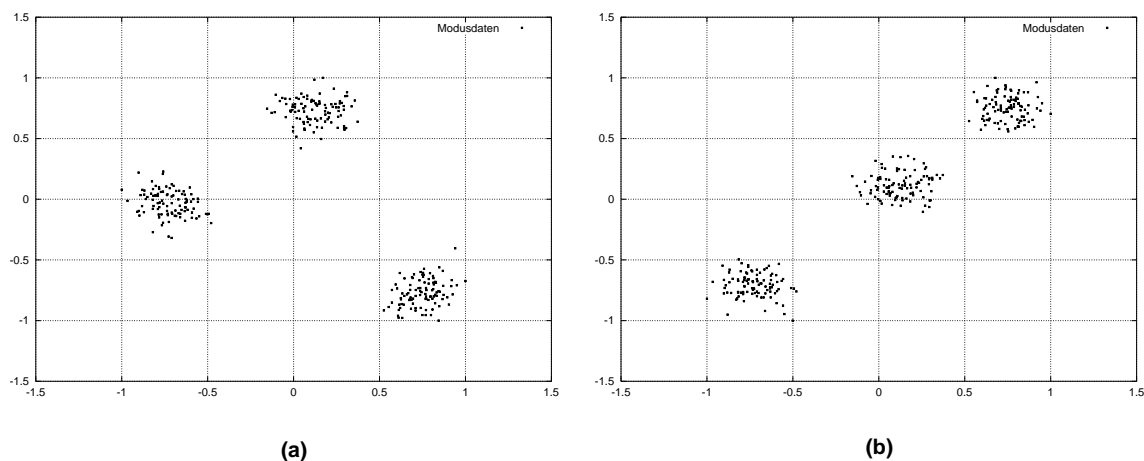


Abb. 2.32.: Zweidimensionale orthogonale Projektionen des 10-dimensionalen Eingabedatensatzes `modes13` (Erläuterung s. Text).

Dem HRCL werden $|D|^{1/3} = 6$ initiale Neuronen mit Radius 0,3 vorgegeben, die zufällig auf Eingabevektoren ξ aus `modes13` gesetzt werden. Das HRCL reduziert im Verlauf des Trainings und ca. 140 Lernschritten (ca. 60 sec) die 6 initialen HRCL-Neuronen zu 3 Neuronen, die so adaptiert werden, daß sie mit den 10-dimensionalen Modizentren des Eingabedatensatzes zusammenfallen. Abb. 2.33. (a) und Abb. 2.33. (b) zeigen jeweils das Ergebnis des HRCL-Trainings als zweidimensionale orthogonale Pro-

jektionen von `modes13`, der adaptierten HRCL-Neuronen, sowie deren projizierten 10-dimensionalen hyperkubischen Umgebungen. Der erwartete Quantisierungsfehler EQE kann von ca. 0.1681 nach 0.1244, der Entropiewert ENT kann von ca. 2.4 nach 0 reduziert werden.

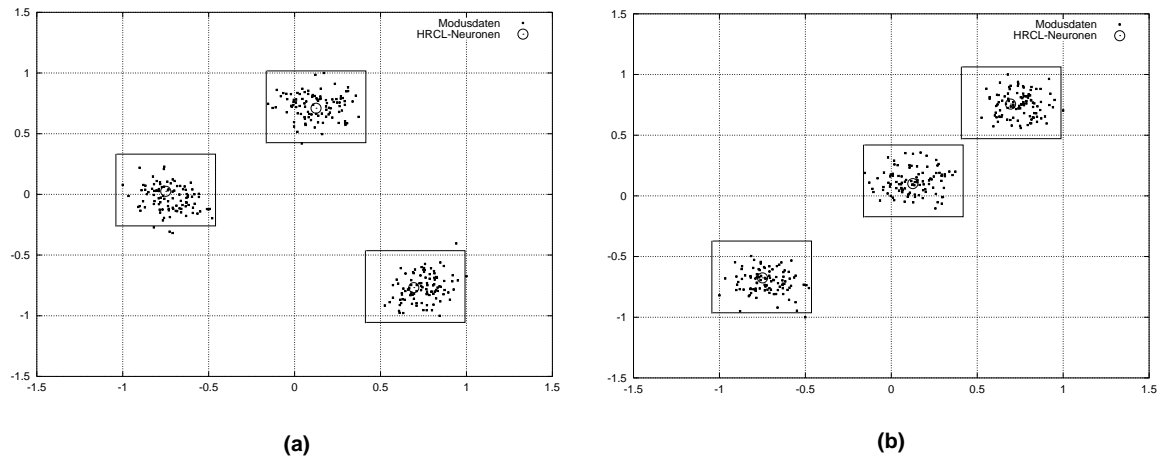


Abb. 2.33.: Ergebnis des HRCL-Trainings als zweidimensionale orthogonale Projektionen von `modes13` und der adaptierten HRCL-Neuronen (vgl. Abb. 2.32.).

- [Bogd98] M. Bogdan: *Signalverarbeitung biologischer Nervensignale zur Steuerung einer Prothese mit Hilfe künstlicher neuronaler Netze*, Dissertation Universität Tübingen, Cuvillier Verlag, Göttingen, 1998
- [DuHa73] R. O. Duda, P. E. Hart: *Pattern Classification and Scene Analysis*, John Wiley & Sons Inc., New York, 1973
- [Fritz97] B. Fritzke: *Some Competitive Learning Methods*, Report of the Systems Biophysics, Institute for Neural Computation, Ruhr-Universität Bochum, April 1997
<http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/Java-Paper>
- [Hamm73] R. W. Hamming: *Numerical Methods for Scientists and Engineers*, McGraw-Hill, 2. Auflage, 1973
- [HeHe95] R. Henrion, G. Henrion: *Multivariate Datenanalyse, Methodik und Anwendung in der Chemie und verwandten Gebieten*, Springer-Verlag, 1995
- [HeRo98] U. Heuser, W. Rosenstiel: *Internetsuche und Neuronale Netze - Stand der Technik*, Research Report WSI 98-10, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, D-72076 Tübingen, Germany, ISSN 0946-3852, 64 pp., 23. September 1998
<http://www-ti.informatik.uni-tuebingen.de/~heuser/papers/rr98.ps.gz>
- [JaDu88] A. K. Jain und R. C. Dubes: *Algorithms for Clustering Data*, Prentice Hall Advanced Reference Series, Englewood Cliffs, New Jersey, 1988
- [Koh82] T. Kohonen: Self-organized Formation of Topology Correct Feature Maps, *Biological Cybernetics*, 43:59-69, 1982
- [Koh84] T. Kohonen: *Self-Organization and Associative Memory*, Springer-Verlag, 1984
- [Koh90] T. Kohonen: The self-organizing map, *Proc. of the IEEE*, Vol. 78, pp. 1464-1480, 1990
- [Koh95] T. Kohonen: *Self-Organizing Maps*, Springer-Verlag, 1995
- [LaWi67] G. N. Lance, W. T. Williams: A general theory of classificatory sorting strategies: II. Clustering systems, in *Computer Journal*, 10, 271-277
- [LoFr97] H. S. Loos, B. Fritzke: *DemoGNG v1.3*, Systems Biophysics at the Institute for Neural Computation, Ruhr-Universität Bochum, 1997
<http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/GNG.html>
<http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/tex/DemoGNG/DemoGNG.html>
- [Reed93] R. Reed: Pruning Algorithms - A Survey, *IEEE Transactions on Neural Networks*, Vol. 4, No. 5, pp. 740-747, September 1993
- [SaMc83] G. Salton, M. J. McGill: *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983
deutsche Übersetzung: G. Salton, M. J. McGill: *Information Retrieval - Grundlegendes für Informationswissenschaftler*, McGraw-Hill, Hamburg, New York, 1987
- [ScEr97] E. Schikuta, M. Erhart: The BANG-Clustering System: Grid-Based Data Analysis, in X. Liu, P. Cohen, M. Berthold (Eds.): *Advances in Intelligent Data Analysis (IDA-97)*,

- LNCS 1280, pp. 513-524, 1997
- [Sedg92] R. Sedgewick, *Algorithms in C++*, Addison-Wesley Publishing Company, 1992
- [Speck95] H. Speckmann: *Analyse mit fraktalen Dimensionen und Parallelisierung von Kohonens selbstorganisierender Karte*, Dissertation, Universität Tübingen, 1995
- [SRR94] H. Speckmann, G. Raddatz, W. Rosenstiel: Relations between generalized fractal dimensions and Kohonen's self-organizing map, *Proc. of Neuro Nimes*, Paris, 1994
<http://www-ti.informatik.uni-tuebingen.de/~speckman/papers/nimes94.ps>
- [Zell+95] A. Zell et. al.: *SNNS: Stuttgart Neural Network Simulator, User Manual, Version 4.1*, University of Stuttgart, Institute for Parallel and Distributed High Performance Systems (IPVR) - Applied Computer Science -- Image Understanding, Report No. 6/95, 1995
<http://www-ra.informatik.uni-tuebingen.de/SNNS/UserManual/UserManual.html>

Anhang A

A.1. Notation

Symbole	Bedeutung
\mathbf{R}^n	n -dimensionaler Vektorraum
$d \in \{1, \dots, n\}$	Dimension des n -dimensionalen Vektorraums
$\mathbf{D} = \{\xi_1, \dots, \xi_M\}, \xi_i \in \mathbf{R}^n$	diskrete Eingabedatenmenge
$M = \mathbf{D} $	Anzahl der Eingabevektoren
$p(\xi), \xi \in \mathbf{R}^n$	kontinuierliche Wahrscheinlichkeitsdichte-Funktion
$\xi, \xi_i \in \mathbf{R}^n$	Eingabevektoren
c_i	Neuron i
$\mathbf{A} = \{c_1, c_2, \dots, c_N\}$	Menge der Neuronen
$N = \mathbf{A} $	Anzahl der Neuronen
$w_{c_i} \in \mathbf{R}^n$	Referenz-/Gewichtsvektor von Neuron c_i
$U_c(r) := (2r)^n$	n -dimensionale hyperkubische Umgebung um Neuron c mit Radius r
$E(\xi)$	Erwartungswert um Vektor ξ als Schätzwert für den statistischen Mittelwert des Clusters
μ	Clustermittelwert
$\Sigma = E((\xi - \mu)(\xi - \mu)^t)$	Kovarianz des Clusters mit Mittelwert μ
s_w	gewinnendes Neuron (Gewinner)

Tabelle 1: Erklärung benutzter Symbole

Symbole	Bedeutung
w_w	Gewicht des Gewinners
$\Theta(r)$	Gleichverteilungsschwelle für Radius r
$w_{c_i}(t)$	Neuronengewicht für Neuron c_i zum Trainingsschritt t
$Z_i, i \in \{0, \dots, 2^n - 1\}$	Zelle i der Zellenclustering
$z_{\min}(d), z_{\max}(d)$	Zellengrenzen einer Zelle für Dimension d
t	Trainingsschritt
t_{\max}	maximale Anzahl von Trainingsschritten
$k_i(\xi, \mathbf{A})$	Ordnung von $c_i \in \mathbf{A}$ relativ zur Distanz von w_{c_i} zur aktuellen Eingabe ξ
$f_{\lambda_1, \lambda_2}(k_i(\xi, \mathbf{A}))$	Adaptions-/Abstoßfunktion bezüglich der Neuronenordnung $k_i(\xi, \mathbf{A})$
$g_{\lambda_3, \lambda_4}(t)$	Adaptions-/Abstoßfunktion bezüglich der Trainingszeit t
$EQE(\rho(\xi), \mathbf{A})$	erwarteter Quantisierungsfehler bezüglich einer kontinuierlichen Signalverteilung $\rho(\xi)$
$EQE(\mathbf{D}, \mathbf{A})$	erwarteter Quantisierungsfehler bezüglich einer diskreten und endlichen Eingabe-Datenmenge \mathbf{D}
V_c	Voronoi-Umgebung um Neuron c
$R_c := \{\xi \mid \ \xi - w_c\ < \ \xi - w_i\ \forall i \neq c\}$	Voronoi-Menge um Neuron c
$NQE(\mathbf{D}, \mathbf{A})$	normierter Quantisierungsfehler
$ENT(\mathbf{D}, \mathbf{A})$	Entropie
$\text{rand}_j(1)$	j -te Zufallszahl zwischen 0 und 1
$(\cdot)^t$	Vektortransposition

Tabelle 1: Erklärung benutzter Symbole

Anhang B

B.1. Ergebnisse SMART

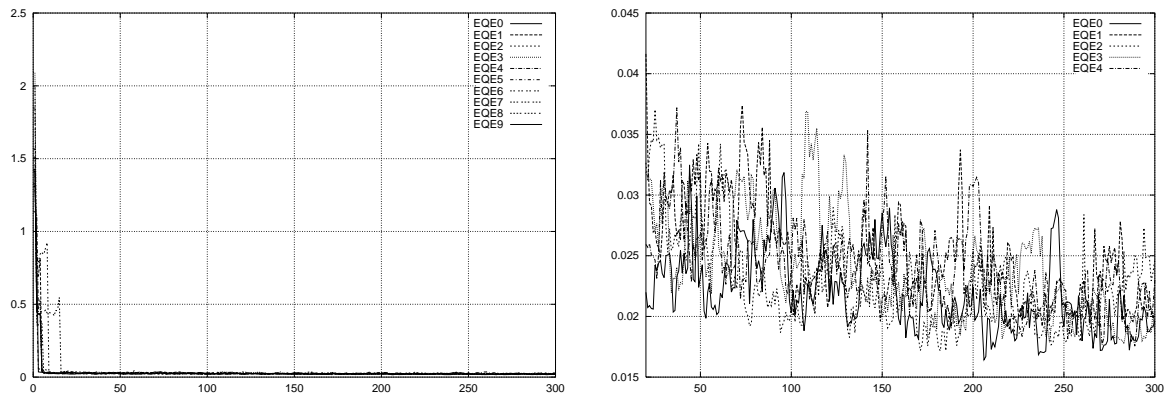


Abb. B.1.: EQE von 10 SMART-Retrievalläufen (rechts eine vergrößerte Darstellung der ersten 5 Läufe ab Adaptionsschritt 20) für `modes1`

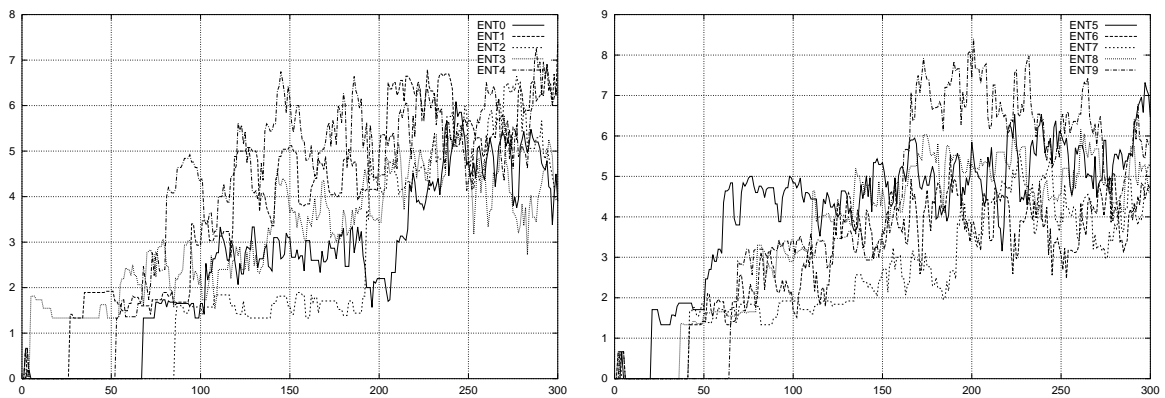


Abb. B.2.: ENT von 10 SMART-Retrievalläufen für `modes1`

B.2. Ergebnisse SOM

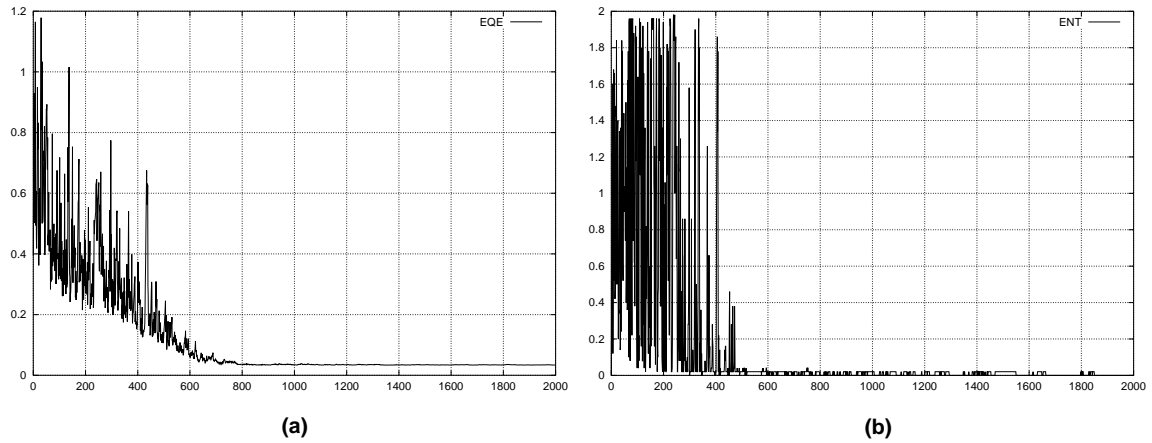


Abb. B.3.: (a) EQE und (b) ENT einer 3x1 SOM für 2.000 Trainingsschritte und `modes1`

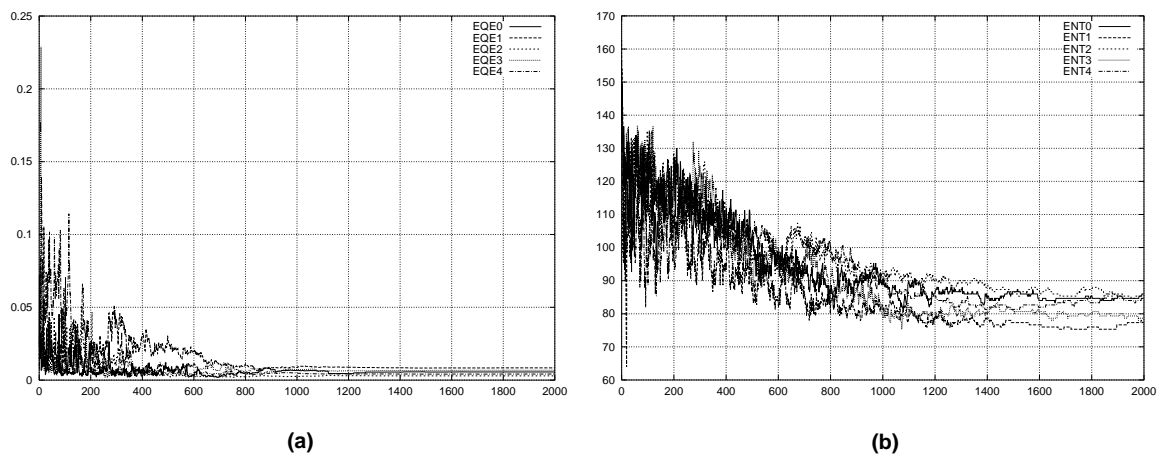


Abb. B.4.: (a) EQE und (b) ENT von 5 10x10 SOM-Läufen a 2.000 Trainingsschritten für `modes1`

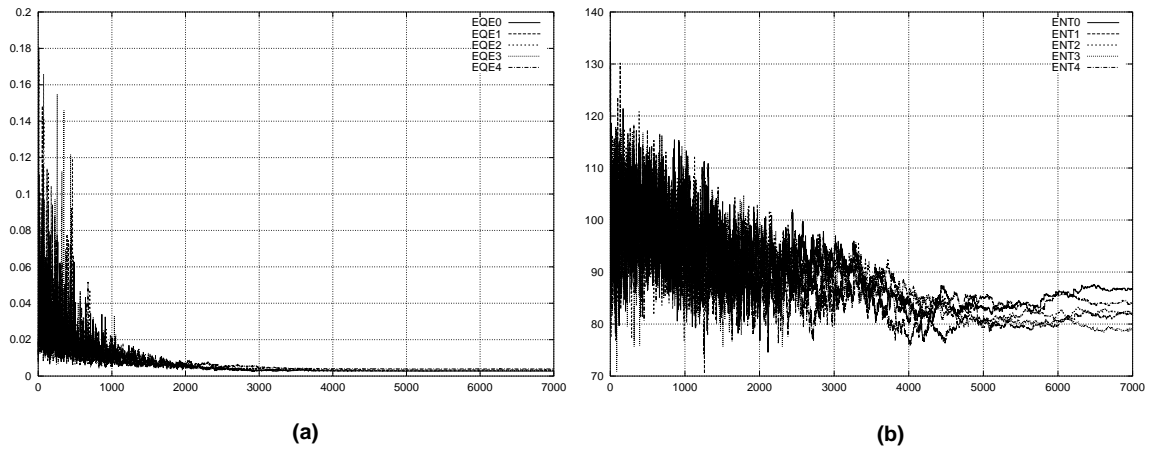


Abb. B.5.: (a) EQE und (b) ENT von 5 10x10 SOM-Läufen a 7.000 Trainingsschritten für modes9

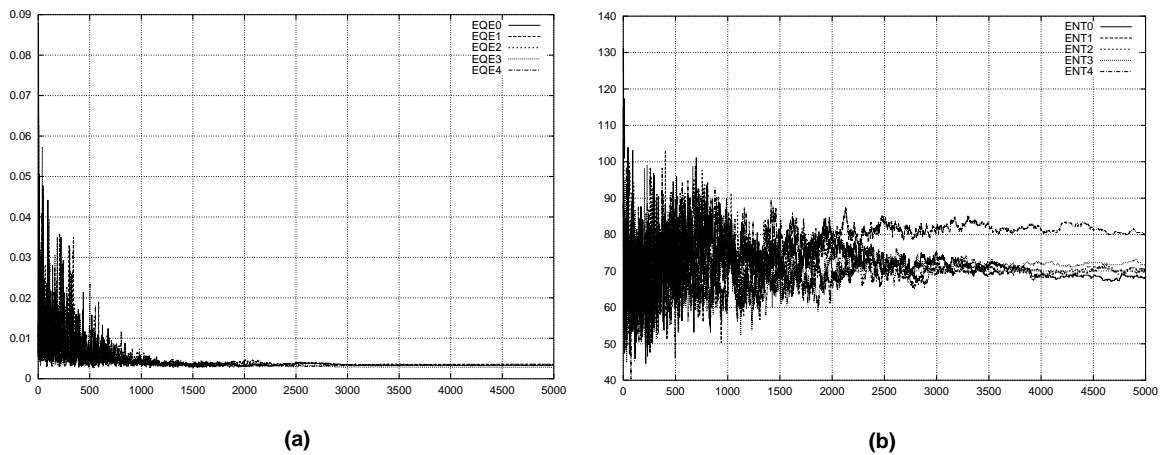


Abb. B.6.: (a) EQE und (b) ENT von 5 10x10 SOM-Läufen a 5.000 Trainingsschritten für modes12

B.3. Ergebnisse Top-Down HRCL

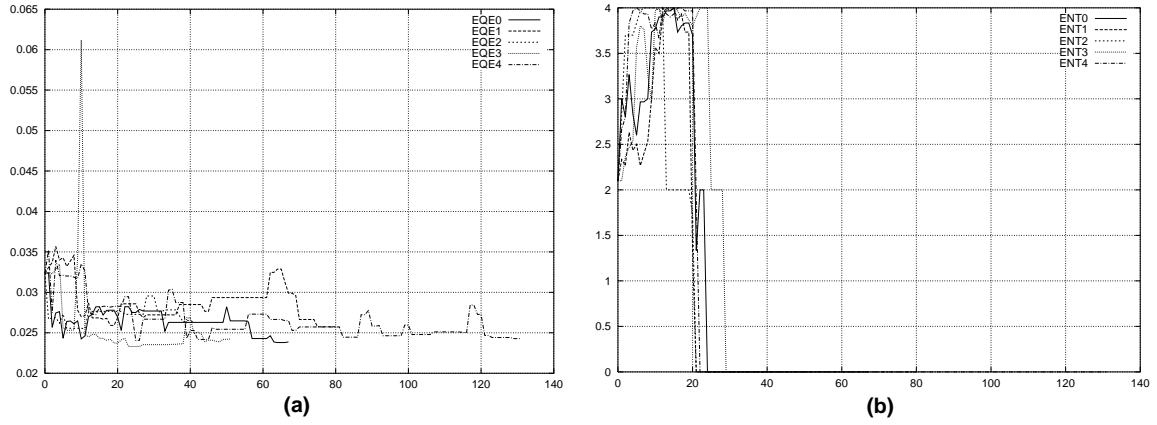


Abb. B.7.: (a) EQE und (b) ENT von 5 HRCL Top-Down Läufen der 0. Hierarchiestufe für modes1

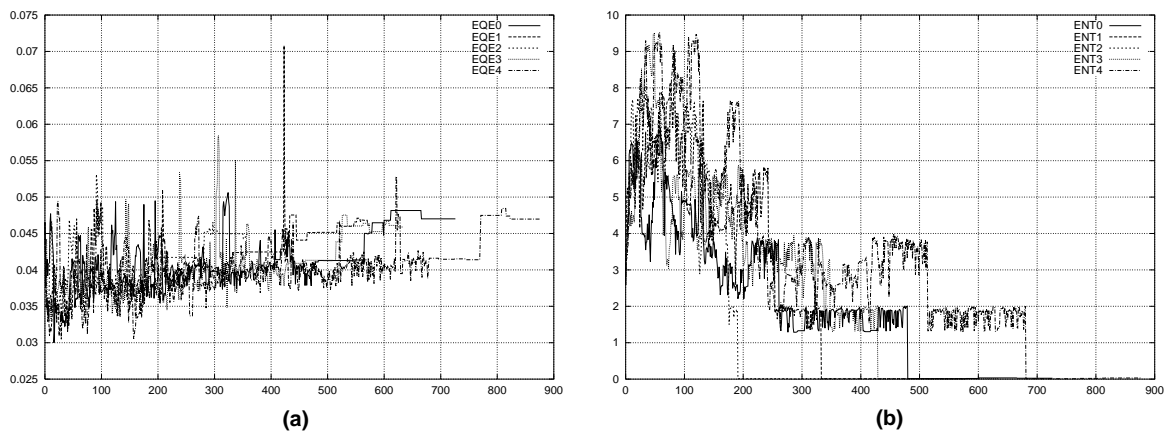


Abb. B.8.: (a) EQE und (b) ENT von 5 HRCL Top-Down Läufen der 0. Hierarchiestufe für modes9

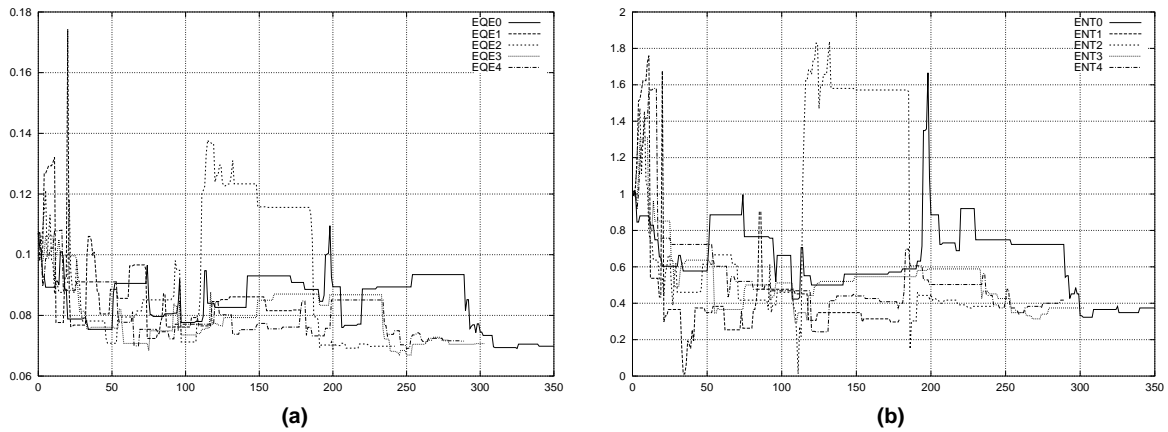


Abb. B.9.: (a) EQE und (b) ENT von 5 HRCL Top-Down Läufen der 0. Hierarchiestufe für modes12

B.4. Ergebnisse Bottom-Up HRCL

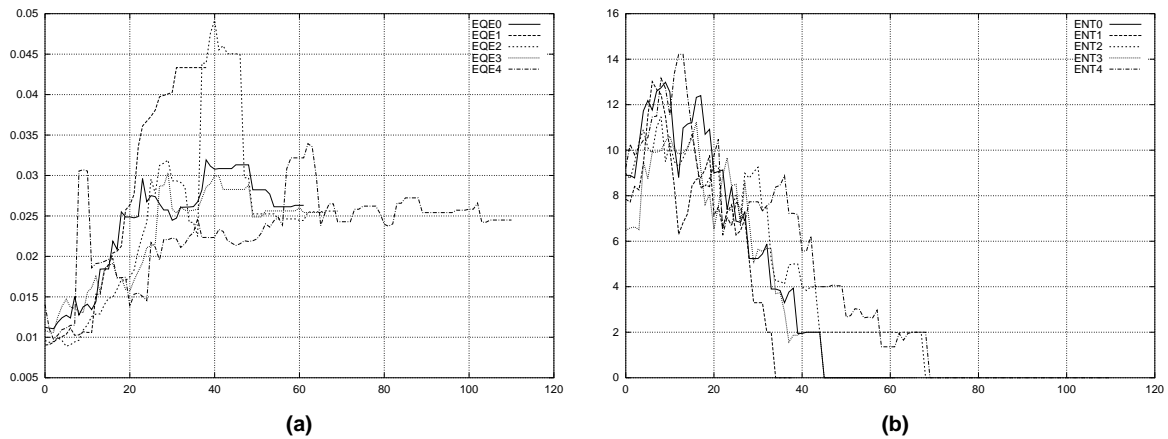


Abb. B.10.: (a) EQE und (b) ENT von 5 HRCL Bottom-Up Läufen der 0. Hierarchiestufe für modes1

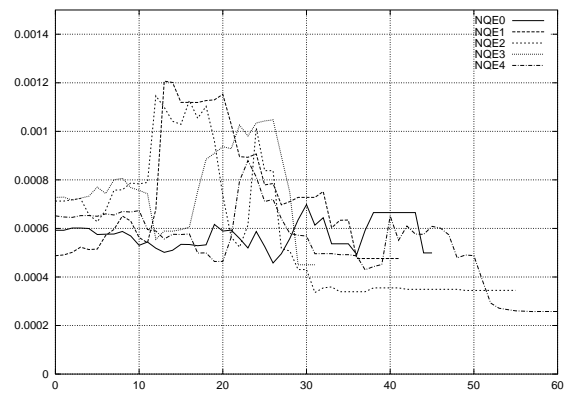
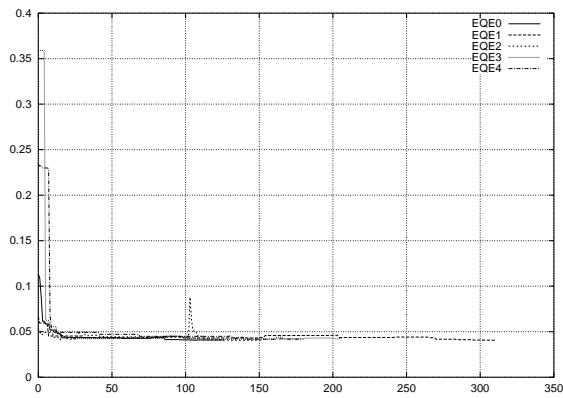
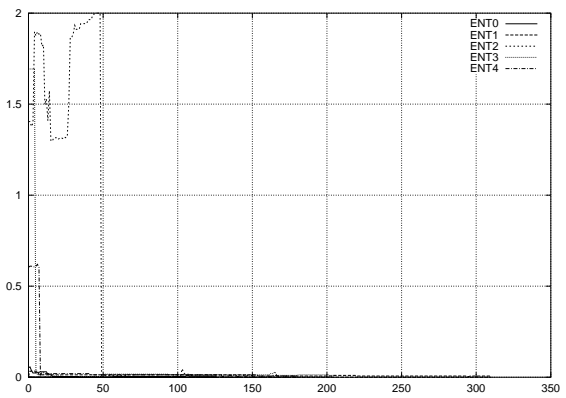


Abb. B.11.: Normierter EQE von 5 HRCL Bottom-Up Läufen der 0. Hierarchiestufe für `modes1`



(a)



(b)

Abb. B.12.: (a) EQE und (b) ENT von 5 HRCL Bottom-Up Läufen der 1. Hierarchiestufe für `modes9`

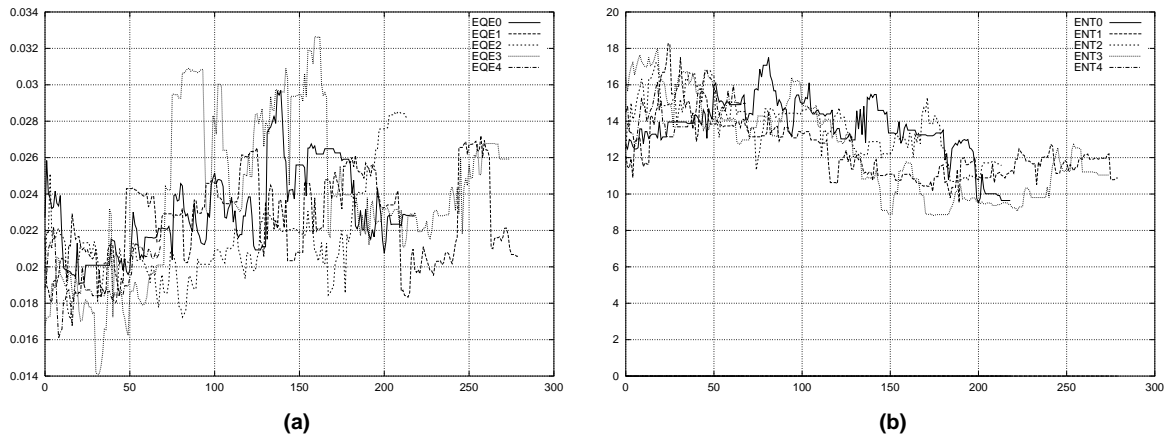


Abb. B.13.: (a) EQE und (b) ENT von 5 HRCL Bottom-Up Läufen der 0. Hierarchiestufe für modes12

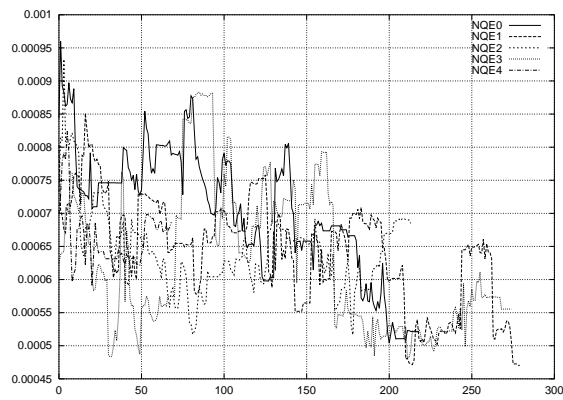


Abb. B.14.: Normierter EQE von 5 HRCL Bottom-Up Läufen der 0. Hierarchiestufe für modes12

