Approximation des TSP mit neuronalen Netzen

Simulation mit self organizing maps

Martin Plaß
Lehrveranstaltung Neuronale Netze
Universität Duisburg-Essen
Wirtschaftsinformatik M.Sc.

Inhalt

ein wenig Neurobiologie (1)

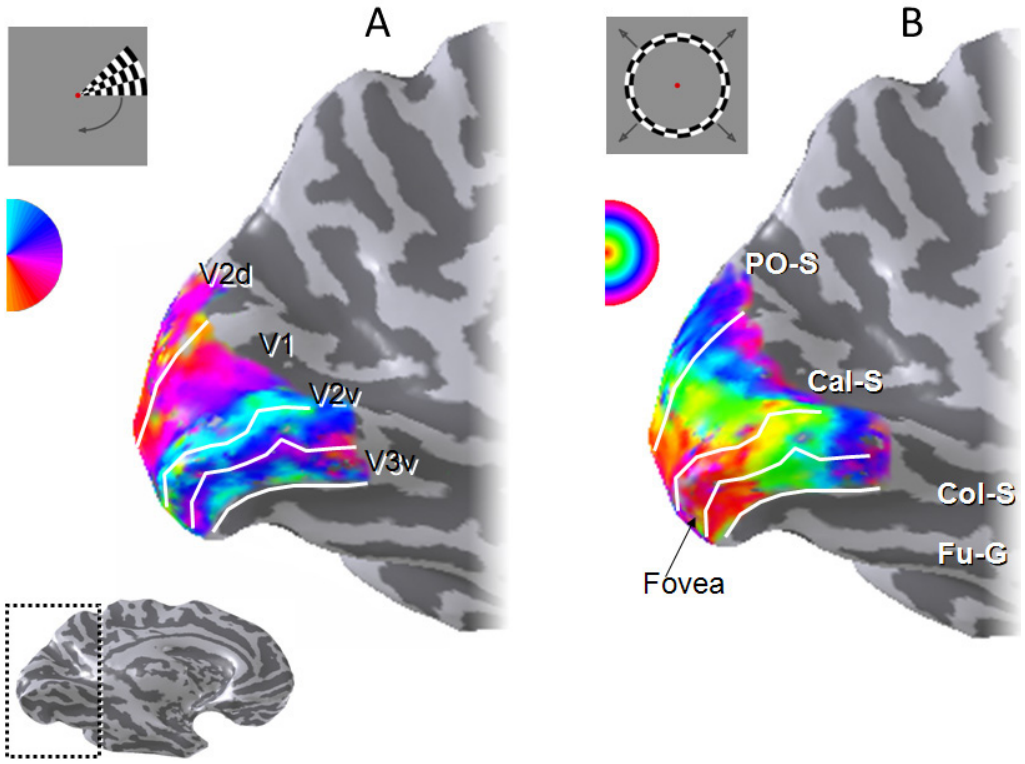
Das kleinste Element im System Gehirn ist das Neuron. Neuronen sind untereinander über Synapsen (quasi Schnittstellen) verschaltet, sodass sich komplexe Stukturen bilden: Neuronale Netze.
Die Funktionsweise von Neuronen ist animiert in diesen youtube-Videos erklärt, wie es besser nicht sein kann (Zugriff jeweils am 23.01.2015):

Neurone: Bausteine des Denkens
— www.youtube.com/watch?v=usosLatcMK8
Elektrische Potenziale: Die Sprache der Neurone
— www.youtube.com/watch?v=1pLNmHSx53w
Synapsen: Schnittstellen des Lernens
— www.youtube.com/watch?v=JONWMYeBIa8

ein wenig Neurobiologie (2)

Sensorische Reize sind mehrdimensional. Wir sehen dreidimensional. Wir können Geräusche im Raum lokalisieren. Der Mensch ist ein dreidimensionaler Körper mitsamt seiner Sensomotorik. Die Informationsverarbeitung im Gehirn findet hingegen in planaren oder linearen Strukturen statt.
Was heißt das für uns? Es findet eine Komplexitätsreduktion bei der Hirnverarbeitung statt. Ein höherdimensionaler Input wird auf einer niederdimensionalen Outputstruktur gemappt – dies ist der Kern: Es geht um Komplexitäts- bzw. Dimensionsreduktion.
Ein Paradebeispiel ist der visuelle Kortex als Teil des okzizeptalen Kortex. Der visuelle Input wird als 2D-Projektion auf dem Cortex abgebildet.

Die rechte Abbildung zeit, wie ein visueller Input im visuellen Kortext abgebildet wird. Es ist zu erkennen, dass die natürliche Anordnung der Farben erhalten bleibt[1]. Die visuellen Signale der Retina sind hier auf die Neuronen des visuellen kortex gemappt (Retinotopie).



1 http://www.intechopen.com/books/visual-cortex-current-status-and-perspectives/visual-field-map-organization-in-human-visual-cortex (Zugriff am 22.01.2015)

ein wenig Mathematik (1)

Es sei eine Reizzentrentopologie V gegeben, bestehend aus einzelnen Reizzentren v_i. Weiterhin sei eine Neuronentopologie N gegeben, welche aus einzelnen Neuronen n_j besteht.
Daher gilt für die Topologien:

$V = \{v_i \mid i = 1..2..p\}$ wobei p die Anzahl der Elemente in V ist.
$N = \{n_j \mid j = 1..2..q\}$ wobei q die Anzahl der Elemente in N ist.

Für den Approximationsalgorithmus sind neben den Topologien V und N folgende Parameter von Relevanz: Der empirische Faktor ψ und die Iterationsanzahl Ω und die Lernrate ε (von Ω abhängig).
Daher kann der Algorithmus als n-Tupel aufgefasst werden:

$ALG_{Approx} = \{V, N, \psi, \Omega, \varepsilon_k \mid k = 1..2..\Omega\}$ wobei k der aktuelle Iterationsschritt ist.

Die Iterationsabhängige Lernrate ε_k berechnet sich wie folgt:

$$\varepsilon_k = \varepsilon_0 \cdot \left[\left(\varepsilon_\Omega \div \varepsilon_0 \right)^{(k \div \Omega)} \right]$$ wobei ε_0 und ε_Ω Konstanten sind und $0 \geq \varepsilon_k \leq 1$.

Damit wird gewährleistet, dass die Lernrate im Kurvenverlauf sinkt. In den anfänglichen Iterationsschritten werden somit hohe Lerneffekte erzielt, während am Ende nur noch kleine Lerneffekte erzielt werden. Die Funktion geht asymptotisch gegen ε_Ω.

ein wenig Mathematik (2)

Als nächstes sei die Auswirkung des empirischen Faktors ψ auf die Nachbarschaftskorrelation innerhalb der Neuronentopologie erläutert.
Ein großer Wert für ψ bewirkt, dass die Neuronen im Lernprozess enger zusammenhängen. Das heißt, dass die Nachbarn des Siegerneurons stärker zum entsprechenden Reizzentrum gezogen werden, als bei einem niedrigen Wert für ψ.

So bewirkt ein Faktor von $\psi = 1{,}0$ eine vollkommene Neuronengewichtung. Das heißt, dass alle Nachbarn (direkte wie auch indirekte) mit der gleichen Gewichtung wie das Siegerneuron in Richtung des gerade aktiven Reizzentrums bewegt werden.
Ein Faktor von $\psi = 0{,}0$ bewirkt hingegen das Gegenteil. Das heißt, dass keine Nachbarn (direkte wie auch indirekte) in Richtung des entsprechenden Reizzentrums bewegt werden.

Für jedes Neuron wird in jedem Iterationsschritt k eine Gewichtung γ berechnet. Ist das Siegerneuron $n_{k|x}$ in k ermittelt, so gilt folgende Berechnungsvorschrift für die jeweilige Gewichtung $\gamma_{k|x\pm i}$ (x wird hier 0 gesetzt und repräsentiert somit das aktuelle Zentrum des Neuronenringes):

$\gamma_{k|x+i} = \gamma_{k|x-i} = \varepsilon_k \cdot \psi^i$ wobei $i = 0..1..2..(q \div 2)$ und $0 \geq \varepsilon_k \leq 1$ und $0 \geq \psi \leq 1$.

Es gibt noch weitere Nachbarschaftsfunktionen (Normalverteilung nach Gauß, Mexican Hat, etc.). Diese hier vorgestellte ist eigenständig empirisch gefunden und erprobt.

ein wenig Pseudocode (1)

Wie wird die Neuronenringtopologie initialisiert?

1 Hole Reizzentrentopologie V & Neuronenanzahl q & Neuronentopologie N

2 Ermittle Schwerpunkt s von V
// arithmetisches Mittel der Koordinaten

3 Ermittle maximale Dimensions-Ausdehnung von V
// entweder Spannweite der x-Dimension oder Spannweite der y-Dimension
Nutze diese als Radius r

4 Setze Winkel a : $a = 0.0$

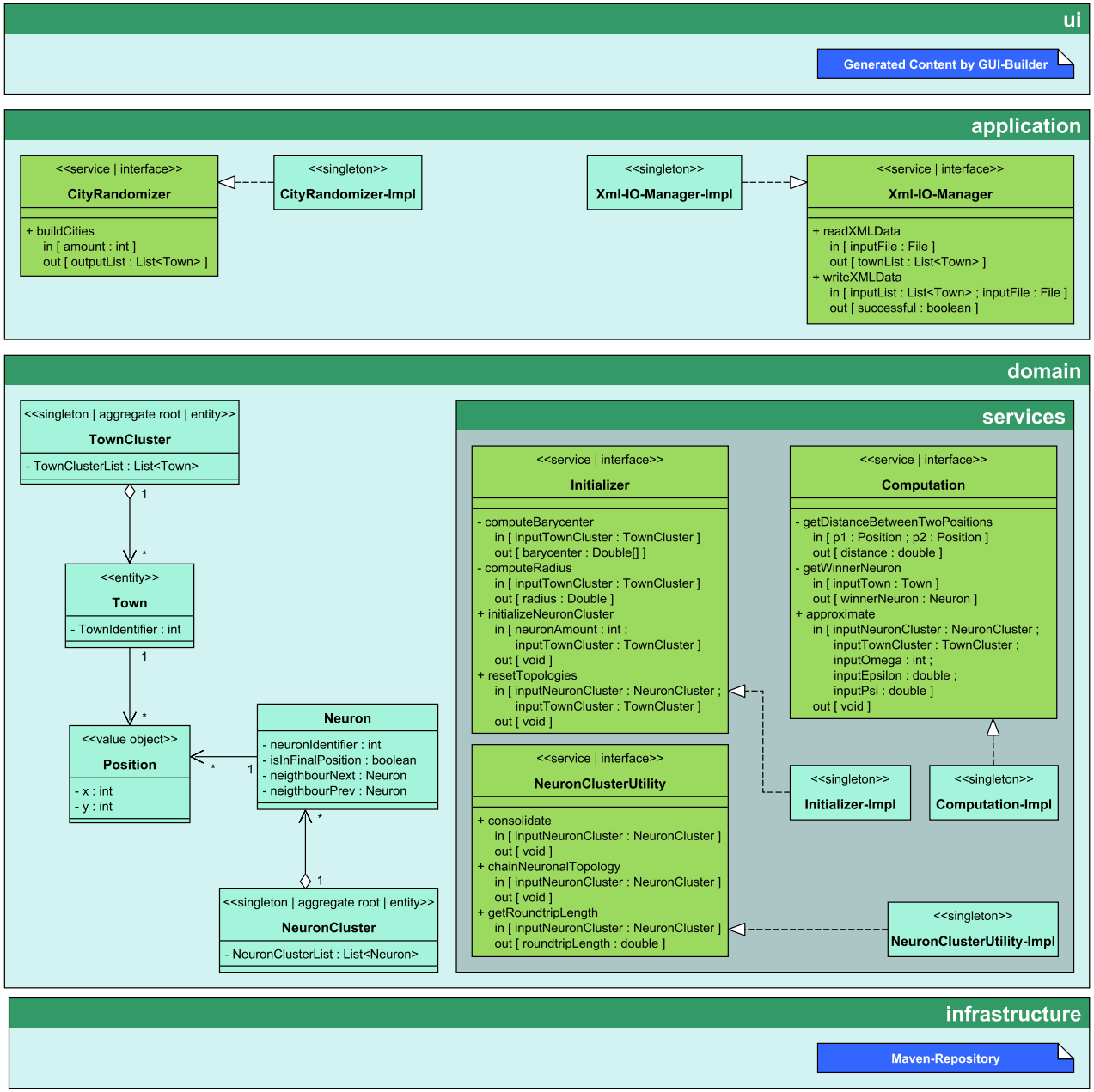
5 Wiederhole q-mal
- 5.1 Erstelle Neuron n
- 5.2 Setze x-Position von n und weise zu : $x = (\sin(a \cdot (\pi / 180)) \cdot r) + s$
- 5.3 Setze y-Position von n und weise zu : $y = (\cos(a \cdot (\pi / 180)) \cdot r) + s$
  // Formeln aus Rotationsmatrix im R2
- 5.4 Füge n zu N hinzu
- 5.5 Berechne a neu und weise zu : $a = a + (360 / q)$

ein wenig Pseudocode (2)

Wie funktioniert die Approximation?

1 Hole Neuronenanzahl q & Neuronentopologie N & Iterationszahl omega & Reizzentrentopologie V & Lernrate epsilon & Empirischer Faktor psi
2 Setze die Konstanten epsilon_s und epsilon_e und weise zu :
epsilon_s = epsilon
epsilon_e = epsilon/100
3 Setze Zählvariable i und weise zu : i = 0
4 Wiederhole omega-mal
4.1 Inkrementiere i : i = i + 1
4.2 Ermittle zufälliges Reizzentrum v aus V
4.3 Ermittle zu v lokales Siegerneuron w
// Neuron mit kleinstem Abstand zu v
4.4 Setze w als aktuelles Neuronenringzentrum und setze den Nachbarschaftsindex f und weise zu : f = 1
4.5 Wiederhole (q/2)-mal
4.5.1 Setze neue Koordinaten des Neurons m mit dem Nachbarschaftsindex f und weise zu :
$x_m = x_m - [x_m - x_v] \cdot epsilon \cdot psi \wedge f$ // epsilon · psi ^ f = gamma
$x_m = x_m - [x_m - x_v] \cdot epsilon \cdot psi \wedge f$ // gamma ist die jeweilige Neuronengewichtung
4.5.2 Inkrementiere f : f = f + 1
4.6 Setze w als erneut als aktuelles Zentrum und setze den Nachbarschaftsindex f und weise zu : f = -1
4.7 Wiederhole (q/2)-mal
4.7.1 Setze neue Koordinaten des Neurons m mit dem Nachbarschaftsindex f und weise zu :
$x_m = x_m - [x_m - x_v] \cdot epsilon \cdot psi \wedge (abs(f))$ // absolutbetrag
$x_m = x_m - [x_m - x_v] \cdot epsilon \cdot psi \wedge (abs(f))$
4.7.2 Dekrementiere f : f = f - 1
4.8 Aktualisiere epsilon und weise zu : epsilon = epsilon_s · [(epsilon_e / epsilon_s) ^ (i / omega)]

Referenzarchitektur



Die Systemarchitektur ist aufgebaut als 4-Schicht-Modell als Empfehlung bei einem Domain Driven Design (DDD)[1].

Die Domainschicht beinhaltet die relevanten Geschäftsobjekte und domänenspezifischen Services.
In der Applikationsschicht sind über die Domäne hinausgehende Features implementiert.
Die UI-Schicht implementiert die vom GUI-Buider generierte GUI.
Die Infrastrukturschicht versorgt die darüberliegenden Systemkomponenten mit den notwendigen Bibliotheken aus dem Maven-Repository.

1 Evans, Eric (2004): Domain Driven Design. Boston: Addison-Wesley.

Ergebnisse der Experimente (1)

Iterationen

Zunächst gilt als globale Aussage, dass das Ausgangsmapping umso feingranulierter ist, je höher die Iterationszahl ist.

Empirischer Faktor

Bei einer geringen Neuronen-Stimulizentren-Ratio sollte der empirische Faktor jedenfalls weniger als $0{,}4$ betragen, da sonst zu wenige Stimulizentren hinreichend approximiert werden (bei einer Ratio von $2{,}0$).
Ein zu geringer empirischer Faktor ist jedoch auch nicht sinnvoll, da in diesem Falle lediglich einzelne Neuronen gereizt werden (Siegerneuron und nur marginal seine Nachbarn).
Daher empfehle ich einen empirischen Faktor von $0{,}15$ bis $0{,}35$ bei einer Ratio von bis zu $2{,}0$ zu wählen. Einen höheren Wert empfehle ich für erste, makroskopische Approximationen, zur Feinjustierung empfehle ich, diesen Faktor zu minimieren.
Weiterhin kann empirisch beobachtet werden, dass bei einer hohen Neuronen-Stimulizentren-Ratio ein kleiner empirischer Faktor dahingehend das Mapping beeinflusst, dass einige Neuronen nicht aktiviert werden, da sie zu weit von Stimulizentren entfernt sind. Es hilft dabei nicht, die Iterationsanzahl zu erhöhen. In diesem Falle ist es sinnvoll, den empirischen Faktor zu erhöhen.

Ergebnisse der Experimente (2)

Bei einer Neuronen-Stimulizentren-Ratio von $4{,}0$ ist dies der Fall. Hier empfehle ich einen empirischen Faktor zwischen $0{,}3$ und $0{,}6$ zu wählen. Zur Feinjustierung ist dann wieder ein kleinerer Wert zu wählen.

Lernrate

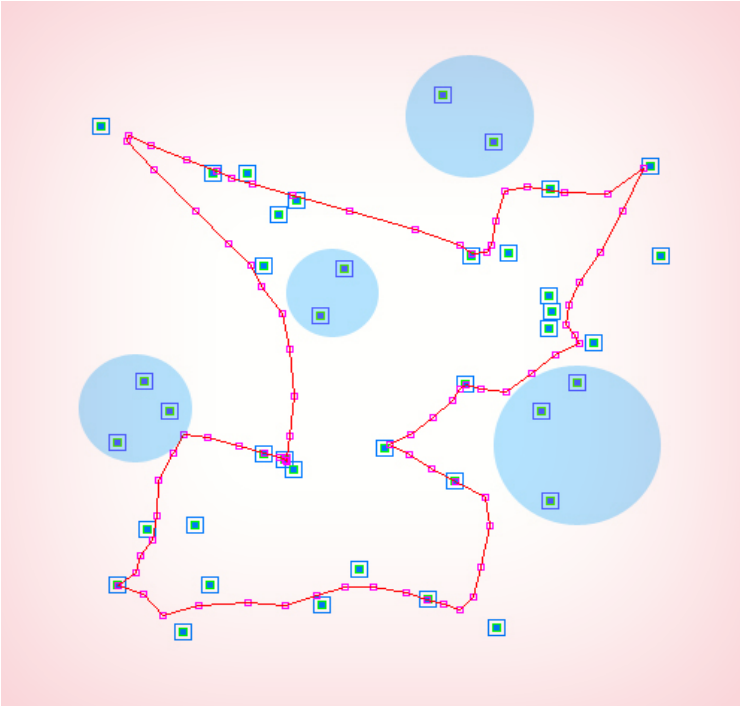
Die Lernrate sollte generell über $0{,}9$ betragen, um eine effektive Neuronenaktivität zu gewährleisten.

Neuronenanzahl

Es sei erwähnt, dass bei relativ dicht gedrängten Städten (sog. Städtehaufen) nicht alle Städte innerhalb dieser Teilmenge konsolidiert werden. Das liegt daran, dass die Neuronendichte zum Zeitpunkt der Konsolidierung in dieser begrenzten Stimuliregion zu gering ist. Es sei daher aus pragmatischen Gründen empfohlen, diesen Städtehaufen im Eingangsmapping zu lockern, d.h. sehr eng beieinanderliegende Städte zu einem Stimulizentrum zusammenzufassen.

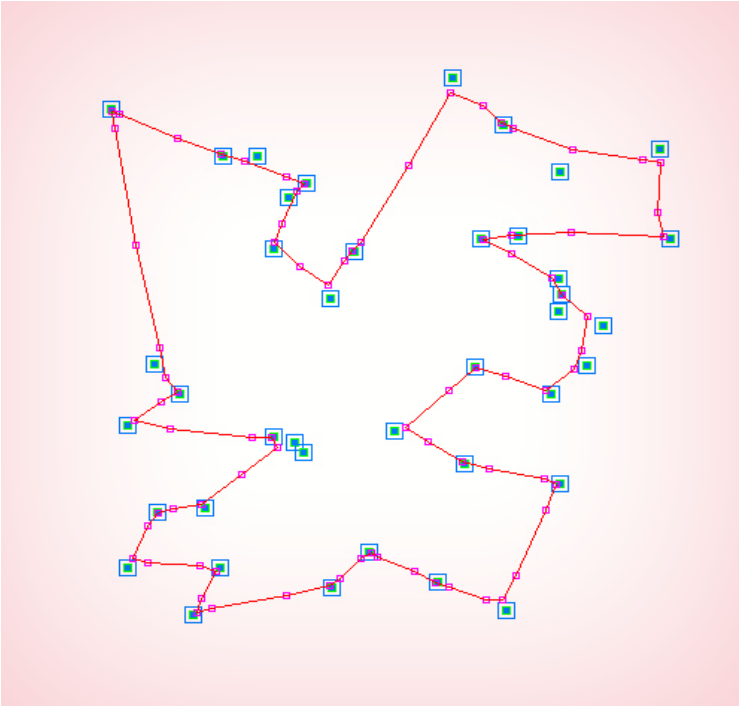
Zu hoher Empirischer Faktor

Manche Städte werden aufgrund der hohen Nachbarschaftskolleration der Neuronentopologie nicht approximiert, da die Neuronen von anderen Reizzentren »weggezogen« werden. Hier hilft eine Erhöhung der Ratio und / oder eine Verringerung des empirischen Faktors.



$q = 80$
$p = 40$
$i = 500$
$\Psi = 0{,}60$
$\varepsilon = 0{,}95$

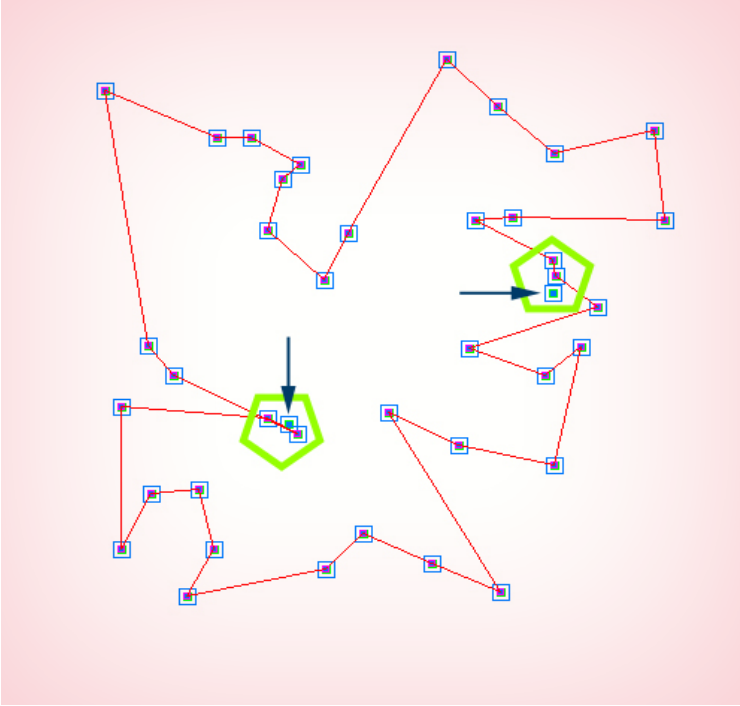
Hier ein Setting mit gleichbleibender Neuronenzahl, aber einem verringertem empirischen Faktor und einer erhöhten Iterationszahl.



$q = 80$
$p = 40$
$i = 2000$
$\Psi = 0{,}30$
$\varepsilon = 0{,}95$

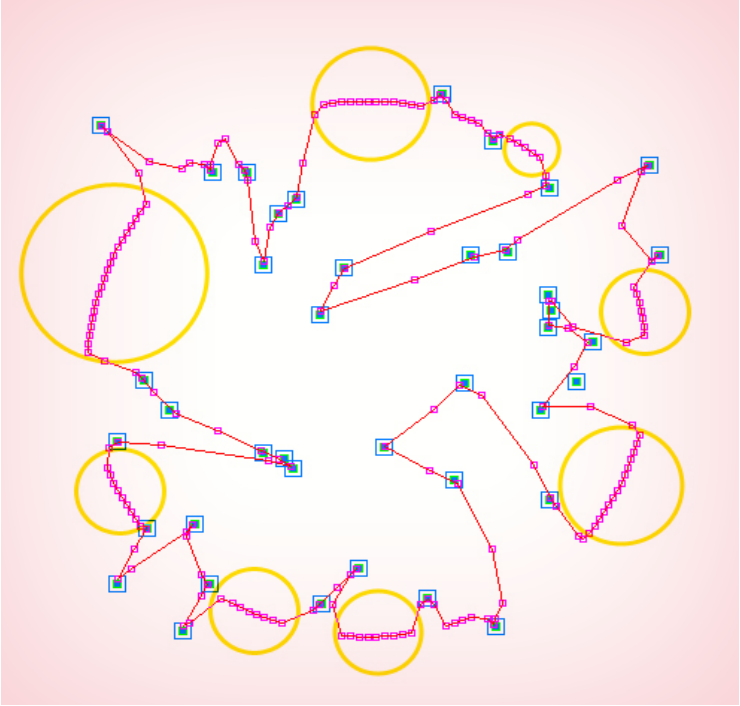
Stadthaufen und tote Neuronen

Städtehaufen – Wird das Setting aus dem letzten Beispiel beibehalten, so kommt es bei der Konsolidierung zu Verfehlungen von Reizzentren aus den genannten Gründen.



$q = 80$
$p = 40$
$i = 2000$
$\Psi = 0{,}30$
$\varepsilon = 0{,}95$

Tote Neuronen – Sie resultieren im Ausgangsmapping, wenn eine hohe Neuronen-Stimulizentren-Ratio gegeben ist und der empirische Faktor zu niedrig gewählt ist.



$q = 160$
$p = 40$
$i = 2000$
$\Psi = 0{,}30$
$\varepsilon = 0{,}95$

Komplexitätsanalyse (1)

Das TSP liegt in der Klasse NP (nichtdeterministisch polynominal); das heißt, dass eine Turingmaschine dieses Problem nicht in polynominaler Zeit lösen kann, da der Lösungsraum überexponentiell wächst. Beim TSP ist es der Fall, dass der Lösungsraum bei steigender Städteanzahl jedenfalls fakultativ wächst, da sich die Anzahl der möglichen Rundreisen in einem symmetrischen Graphen ab einer Städteanzahl wie folgt berechnet:

$A_n = (n-1)! \div 2$ mit A Anzahl der Elemente im Lösungsraum und n Anzahl der Städte.

Bei 10 Städten sind 181.440 Rundreisen möglich. Bei 20 sind es schon 60.822.550.204.416.000 und weil es so schön ist: bei 40 Städten sind es gar 10.198.941.040.598.700.000.000.000.000.000.000.000.000.000.000 .
Eine optimale Lösung in einer praktisch zumutbaren Zeit zu finden, ist also nicht gegeben. Die Problematik des TSP lässt sich daher in der Komplexitätsklasse $\mathcal{O}(n!)$ verorten, wenn das TSP nach dem klassischen graphentheoretischen Paradigma perspektiviert ist.
Der Ansatz mit SOMs perspektiviert das TSP grundlegend anders. Es wird nicht innerhalb der Stadttopologie, bzw. des Stadtgraphen nach einer Lösung gesucht, sondern die Lösung nähert sich gewissermaßen an die Stadttopologie organisch-agil heran. Während in graphentheoretisch motivierten Algorithmen der Städtegraph ein aktives Element der Berechnung darstellt, so nimmt er in dem hier vorgestellten Algorithmus die passive Rolle eines Attraktors ein, welcher gewissermaßen eine Hilfstopologie für ihn arbeiten lässt – und das innerhalb einer weitaus geringeren Komplexitätsklasse, was im Folgenden empirisch untermauert wird.

Komplexitätsanalyse (2)

Iterationen	200 Neuronen	400 Neuronen	Neuronen-Ratio	Iterations-Differenz bei 200 N	Iterations-Differenz bei 400 N	Differenz-Ratio
1000	142	300	2,112676			
2000	297	594	2,000000	155	294	1,896774
3000	427	874	2,046838	130	280	2,153846
4000	579	1155	1,994819	152	281	1,848684
5000	719	1451	2,018081	140	296	2,114286
6000	843	1717	2,036773	124	266	2,145161
7000	948	1982	2,090717	105	265	2,523810
8000	1125	2278	2,024889	177	296	1,672316

Links oben und rechts unten Testsettings mit einer Reizzentrenanzahl von 40. Links unten eines mit 80. Die Daten weisen auf ein lineares Wachstum der Laufzeit hin, dessen Maß von der Neuronenanzahl und der Iterationsanzahl determiniert ist.

Somit lässt sich die Komplexität in $\mathcal{O}(n)$ verorten. Die Laufzeit lässt sich einheitenlos prognostizieren: $L = q \cdot \Omega$

mit q Neuronenanzahl und Ω Iterationsanzahl.

Iterationen	200 Neuronen	400 Neuronen	Neuronen-Ratio	Iterations-Differenz bei 200 N	Iterations-Differenz bei 400 N	Differenz-Ratio
1000	142	296	2,084507			
2000	281	562	2,000000	139	266	1,913669
3000	421	874	2,076010	140	312	2,228571
4000	561	1134	2,021390	140	260	1,857143
5000	717	1435	2,001395	156	301	1,929487
6000	858	1700	1,981352	141	265	1,879433
7000	982	1997	2,033605	124	297	2,395161
8000	1124	2277	2,025801	142	280	1,971831

Iterationen	200 Neuronen	400 Neuronen	Neuronen-Ratio	Iterations-Differenz bei 200 N	Iterations-Differenz bei 400 N	Differenz-Ratio
5000	1795	3574	1,991086			
10000	3558	7163	2,013210	1763	3589	2,035735
15000	5352	10703	1,999813	1794	3540	1,973244
20000	7146	14278	1,998041	1794	3575	1,992754
25000	8893	17849	2,007084	1747	3571	2,044076
30000	10687	21451	2,007205	1794	3602	2,007804
35000	12435	25000	2,010454	1748	3549	2,030320
40000	14229	28565	2,007520	1794	3565	1,987179

Bibliographie

Zur verwendeten Softwarearchitektur:

Evans, Eric (2003)
Domain Driven Design: Tackling Complexity in the Heart of Software. Boston: Addison-Wesley.

Zum biologischen Hintergrund:

Jäger, Marten (2007)
Selbstorganisierende Karten.
Berlin, Humboldt Universität zu Berlin, Seminararbeit zur theoritischen Biologie.
URL: http://jaguar.biologie.hu-berlin.de/~wolfram/pages/seminar_theoretische_biologie_2007/ausarbeitungen/jaeger.pdf
(Zugriff am 22.01.2015).

Hoffmann, Michael B. et al. (2011)
Retinotope Kartierung des menschlichen visuellen Kortex mit funktioneller Magnetresonanztomografie – Grundlagen, aktuelle Entwicklungen und Perspektiven für die Ophthalmologie
URL: http://www.med.uni-magdeburg.de/augenklinik/vpl/papers_pdfs/2011%20HoffmannKliMo_retinotopic_mapping.pdf
(Zugriff am 22.01.2015).

Zur Komplexitätstheorie:

Grumm, Heinz-Peter und Sommer, Manfred (2010)
Einführung in die Informatik. München: Oldenbourg Wissenschaftsverlag.

Weiterführende Links zur Forschung im vorgestellten Bereich

Die AG Datenbionik der Fakultät Mathematik und Informatik der Universität Marburg beschäftigt sich mit der Ordnung von Strukturen durch naturanaloge Algorithmen.
www.uni-marburg.de/fb12/datenbionik/forschung/index_html/ (Zugriff am 22.01.2015)

die Forschungsgruppe COBASC an der Universität Duisburg-Essen beschäftigt sich unter anderem mit der Analyse komplexer sozioökonomischer Systeme und semantischen Netzwerken durch Methoden des Soft Computings.
www.cobasc.de/softcomputing/content/view/12/26/ (Zugriff am 22.01.2015)