Kapitel 7

Selbstorganisierende Karten

Wir betrachten nun eine wichtige Klasse von neuronalen Netzen, die als Modelle neuronaler Karten dienen. Karten stellen ein Hauptelement der kortikalen Informationsverarbeitung dar. Von verschiedenen Autoren wird die gesamte Informationsverarbeitung im Kortex als eine Hierarchie von Karten in Stufen steigender Abstraktion der primären Sinneseindrücke (sensorische Inputs) betrachtet (van Essen). Auf jeden Fall spielen Karten eine wichtige Rolle beim Aufbau einer inneren Repräsentation der Außenwelt. Diese Repräsentation ist immer informationsreduziert, d. h. es findet immer eine Datenreduktion statt. Jedes Neuron, das an der Repräsentation eines primären Sinneseindruckes beteiligt ist, überdeckt mit seinem rezeptiven Feld einen Teil der sensorischen Informationsflächen und ist damit für eine bestimmte Auswahl der sensorischen Reize "zuständig".

Im folgenden werden wir uns zunächst mit der datenreduzierenden Projektion der sensorischen Inputs auf die zuständigen Neuronen beschäftigen. Technisch führt das auf den Begriff des Vektorquantisierers und auf die Frage, wie eine solche Datenreduktion möglichst effektiv, d. h. mit möglichst geringen Informationsverlusten, geleistet werden kann. Wie sich zeigen wird, kann das mit einem einfachen Wettbewerbslernen biologisch effektiv realisiert werden. Danach soll die Organisation der Einzelprojektionen auf die Neuronen in eine topografische Abbildung untersucht und insbesondere die Frage studiert werden, wie diese topografischen Abbildungen gelernt werden können.

7.1 Der Vektorquantisierer

7.1.1 Rezeptive Felder und Voronoiparkettierung des Inputraumes

Wir betrachten eine Menge von Neuronen $i = 1, ..., N$, die alle von den n Inputunits $l = 1, ..., n$ gespeist werden. Die Inputvektoren (Merkmalsvektoren) $\vec{x} = (x_1, ..., x_n) \in \mathcal{R}^n$ werden als Punkte im n - dimensionalen Raum gedacht.

Jedes Neuron i führt den Vektor $\vec{w}_i = (w_{i1}, ..., w_{in})$ seiner synaptischen Gewichte, der ebenfalls als Punkt im $\mathcal{R}^n$ interpretiert werden kann. Die Menge der $\{\vec{w}_i\}$ definiert eine Voronoiparkettierung des $\mathcal{R}^n$ d. h. eine Zerlegung des Raumes in die Menge $\mathcal{V} = \{V_i \mid i = 1, ..., n\}$ der Voronoizellen der einzelnen Vektoren $\vec{w}_i$, so daß

$$\mathcal{R}^n = V_1 \cup V_2 \cup ... \cup V_N$$

Sei im $\mathcal{R}^n$ ein Abstand $D(\vec{x}, \vec{y})$ zwischen zwei beliebigen Vektoren $\vec{x}$, $\vec{y}$ definiert. Das kann z. B. der Euklidische Abstand

$$D(\vec{x}, \vec{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

sein. Die zum Vektor $\vec{w}_i$ gehörige Voronoizelle V_i ist definiert als das Gebiet aller $\vec{x}$, die zu $\vec{w}_i$ einen kleineren (oder höchstens den gleichen) Abstand als zu allen restlichen Repräsentanten haben

$$V_i = \{\vec{x} \mid D(\vec{x}, \vec{w}_i) \leq D(\vec{x}, \vec{w}_j) \quad \forall j \neq i\} \tag{7.1}$$

Die Grenzen der einzelnen Voronoizellen sind folglich Hyperebenen, die die Verbindungslinien zwischen benachbarten Repräsentanten auf halbem Wege senkrecht schneiden.

Betrachtet man die Übereinstimmung des Inputs mit dem synaptischen Vektor als Maß für die Erregung des Neurons, so ist die Voronoizelle das Gebiet der Inputs, die das betrachtete Neuron i am stärksten erregen. Dieses wird deshalb für gegebenes $\vec{x}$ auch als das *Siegerneuron* oder best match Neuron bezeichnet. Die Voronoizelle ist damit die Domäne des Neurons und ist gleichbedeutend mit seinem rezeptiven Feld.[1]

7.1.2 Der Vektorquanisierer (VQ)

VQ werden allgemein zur **Datenreduktion** eingesetzt, d. h. zur problemangepaßten Diskretisierung (=Quantisierung) oder Partitionierung von hochdimensionalen Datenräumen, für die bei der Kodierung ein gewisser Informationsverlust hingenommen werden kann. (Unterschied zur **Datenkompression** bei der eine vollständige Rekonstruktion der Ausgangsdaten möglich sein soll). Ziel ist, eine diskrete Menge von Datenvektoren $\vec{x}$ in Gruppen zusammenzufassen, die jeweils durch einen einzigen Vektor, ihren **Codebuchvektor** oder **Repräsentanten**, repräsentiert werden. Das entspricht der Aufteilung der Menge der Datenvektoren in Teilmengen entsprechend den Gruppen $i = 1, 2, \ldots, N$, wobei allen $\vec{x} \in$ Gruppe i der Repräsentant $\vec{w}_i$ zugeordnet wird (**Kodierungsschritt**). Umgekehrt ist die Gruppierung durch die Voronoiparkettierung des Raumes in eindeutiger Weise

[1] In der neurobiologischen Formulierung können die rezeptiven Felder durchaus auch überlappend sein.

vorgegeben. Zur Informationsübertragung wird dann nur der Index i übertragen und im Decodierungsschritt dann $\vec{w}_i$ ausgegeben

$$\text{Input} \quad \vec{x} \quad \Rightarrow \quad \boxed{\begin{array}{c}\text{Kodierer}\\ \vec{x} \;\rightarrow\; i\end{array}} \quad \rightarrow \quad i \quad \rightarrow \quad \boxed{\begin{array}{c}\text{Dekodierer}\\ i \;\rightarrow\; \vec{w}_i\end{array}} \quad \Rightarrow \; \vec{w}_i \quad \text{Output} \tag{7.2}$$

Der Kodierer ist also durch die Funktion $i:\ \vec{x} \rightarrow i\left(\vec{x}\right)$ gegeben, wobei (s. 7.1)

$$i = \arg\min_j D\left(\vec{x}, \vec{w}_j\right) \tag{7.3}$$

d. h. die Datenvektoren zum Repräsentaten $\vec{w}_i$ liegen alle in dessen Voronoizelle.

Frage ist nun, wie eine möglichst effektive Vekrorquantisierung aussieht. Zur Beurteilung der Güte einer Vektorquantisierung gibt es im wesentlichen zwei Kriterien – die übertragene Transinformation und den Rekonstruktionsfehler.

Informationstheoretische Betrachtungen

Der VQ kann als ein Kodierer betrachtet werden, der Inputs $\vec{x}$ Klassenlabel $l, \quad l = 1, \ldots, L$ zuordnet. Wir stellen nun zunächst die Frage nach der im Sinne der Informationstheorie optimalen Form der Kodierung. Diese Frage kann ganz allgemein ohne Bezug auf eine spezielle Realisierung eines VQ beantwortet werden. Die geeignete Größe ist die Transinformation, die die durch den Kodierer übertragene Information mißt. Wenn die Inputvektoren $\vec{x} \in \Re^n$ stochastische Vektoren mit einer Verteilung $p\left(\vec{x}\right)$ sind, so ist die Inputinformation

$$I^{(in)} = -\sum_{\vec{x}} p\left(\vec{x}\right) \ln p\left(\vec{x}\right)$$

Wegen der Zuordnung $\vec{x} \rightarrow \vec{w}_l$ bzw. $\vec{x} \rightarrow$Klassenlabel $l, \quad l = 1, \ldots, L$ definiert die Verteilung $p\left(\vec{x}\right)$ auch eine output Verteilung $P\left(l\right)$ mit Information

$$I^{(out)} = -\sum_{l=1}^{L} P\left(l\right) \ln P\left(l\right)$$

Die bedingte output Information $I^{(out|\vec{x})} = -\sum_{l=1}^{L} P\left(l \mid \vec{x}\right) \ln P\left(l \mid \vec{x}\right)$ bzw. im Mittel über die Inputs

$$I^{(out|in)} = \left\langle I^{(out|\vec{x})} \right\rangle = \sum_{\vec{x}} p\left(\vec{x}\right) \sum_{l=1}^{L} P\left(l \mid \vec{x}\right) \ln P\left(l \mid \vec{x}\right)$$

ist ein Maß für die output Information bei festgehaltenem Input $\vec{x}$. Bei einer (wie im betrachteten Fall) deterministischen Zuordnung ist $P\left(l \mid \vec{x}\right)$ entweder Null oder Eins und damit ist die bedingte Information gleich Null, wie es sein muß, denn die bedingte Information mißt die durch das Rauschen des Kodierers erzeugte "falsche" Information.. Die am Ausgang anliegende nutzbare oder

unverfälschte Information ist gerade die Transinformation, d. h. die durch den Kodierer übertragene Information

$$I^{(trans)} = I^{(out)} - I^{(out|in)}$$

Im betrachteten Fall gilt also $I^{(trans)} = I^{(out)}$ und der optimale Kodierer ist durch $I^{(out)} = \max$ gegeben, was durch

$$P(l) = \frac{1}{L}$$

erfüllt wird, d., h. **der Kodierer arbeitet optimal, wenn alle Repräsentanten gleich oft angesprochen werden**, wenn also alle Voronoizellen im Mittel gleich viele Datenpunkte enthalten. Für $p(\vec{x}) = const$ heißt das, daß alle Domänen gleich groß sind.

Der Rekonstruktionsfehler

Der Rekonstruktionsfehler drückt den durch die Ersetzung der $\vec{x}$ durch die $\vec{w}$ gemachten Fehler in Termen der mittleren quadratischen Abweichung aus

$$E(\mathbf{w}) = E(\{w_i \mid i = 1, \ldots, N\}) = \langle E(\mathbf{w}, \vec{x}) \rangle = \sum_{\vec{x}} p(\vec{x})\, E(\mathbf{w}, \vec{x})$$

mit

$$E(\mathbf{w}, \vec{x}) = D\left(\vec{w}_{i(\vec{x})}, \vec{x}\right)^2$$

wobei $i(\vec{x})$ der Index des Siegervektors zu $\vec{x}$ ist, d. h. $D\left(\vec{w}_{i(\vec{x})}, \vec{x}\right) = \min_l D(\vec{w}_l, \vec{x}) \quad \forall \vec{x}$.

Bei gleichverteilten Daten ist der Rekonstruktionsfehler am kleinsten, wenn alle Zellen gleich groß sind. Das enspricht also auch dem Maximum der Transinformation. Bei inhomogenen Datenverteilungen liefern beide Maße aber durchaus unterschiedliche optimale VQ.

7.1.3 Wettbewerbslernen

Die Bestimmung des besten Satzes von Prototypvektoren erfordert die Minimierung von E bzw. die Maximierung von I^{trans} als Funktion der $\vec{w}$. Das kann durch einen Gradientenfolgeverfahren erreicht werden. Für die Minimierung des Rekonstruktionsfehlers E wird dabei jeder Vektor $\vec{w}_i$ schrittweise gemäß

$$\Delta \vec{w}_{i\alpha} = -\epsilon \frac{\partial E}{\partial w_{i\alpha}} \tag{7.4}$$

verändert verändert.

Dazu muß aber der mittlere Fehler E bekannt sein, d. h. man muß schon alle Daten vorliegen haben. Ein inkrementelles Verfahren, bei dem mit jedem

neu einkommenden Datenvektror $\vec{x}$ eine "infinitesimal" kleine Veränderung der vorgenommen wird, ergibt sich durch die stochastische Approximation des Gradientenabstieges 7.4. Das wird mit folgendem Algorithmus realisiert: Bei Eingabe von $\vec{x}$ bestimmen wir zunächst das "Siegerneuron" i, d. h. den *best match* Repräsentanten $\vec{w}_i$) und führen für dieses den Lernschritt gemäß

$$\Delta\vec{w}_i = -\epsilon\,(\vec{w}_i - \vec{x}) \qquad i = i\,(\vec{x}) \tag{7.5}$$

aus, mit geeigneter Abkühlungsregel für den Lernparameter ϵ.(Linde, Buzo, Gray 1980, s. [14]). Der Algorithmus erzeugt einen zeitdiskreten stochastischen Prozeß $\vec{w}\,(t)$. Im stationären Zustand gilt $\langle\Delta\vec{w}\rangle = 0$ und folglich

$$\langle\vec{w}_l\rangle^{(l)} = \langle\vec{x}\rangle^{(l)}$$

wobei die $\langle\vec{x}\rangle^{(l)}$ die Mittelung über die Domäne des Repräsentanten $\vec{w}_l$ ist. Der Repräsentant ist also bei sorgfältigem "Einfrieren" gleich dem Mittelwert der in seiner Domäne gelegenen Inputvektoren $\vec{x}$.

Das Verfahren heißt Wettbewerbslernen, weil die Neuronen im Wettbewerb um die beste Repräsentation der Inputs stehen.

7.1.4 Der überwacht lernende Vektorquantisierer (LVQ)

Das oben eingeführte Wettbewerbslernen als Verfahren zur Optimierung eines Vektorquantisierers ist ein nichtüberwachtes Lernverfahren. In der Tat wird kein Information durch einen Lehrer vorgegeben, die Positionierung der Repräsentanten und damit die Gruppierung der Inputvektoren erfolgt rein nach statistischen Gesichtspunkten.

Der LVQ als Klassifikator

Ein Klassifikator teilt die Inputvektoren in L **vorgegebene** Klassen ein. Der LVQ ist ein lernender Klassifikator, der anhand von Beispielen lernt, die Inputvektoren in genau diese Klassen einzuteilen. Die Klassen belegen bestimmte Gebiete im Raum der Inputvektoren. Diese Gebiete müssen mit Repräsentanten der jeweiligen Klasse belegt werden. Für eine möglichst variable Grenzziehung zwischen den Gebieten der verschiedenen Klassen müssen die Klassen jeweils durch hinreichend viele Prototypvektoren repräsentiert werden. Zu jeder Klasse Γ_k gehört folglich eine Menge von klassenspezifischen Repräsentanten $\vec{w}_{k_l}$, $l = 1, \ldots, n_k$. Zur Klassifikation wird $\vec{x}$ genau dann der Klasse zugeordnet, wenn es **einem** der Repräsentanten dieser Klasse den kleinsten Abstand hat.

Ziel des Lernverfahrens ist es, die Repräsentanten $\vec{w}_{k_l}$ zu lernen. Ein gut geeignetes Verfahren wurde von *Kohonen* [12] angegeben und *Lernender Vektorquantisierer* (LVQ) genannt. In der Grundversion lautet der Lernalgorithmus des LVQ für eine Euklidische Abstandsfunktion und für diskrete Zeiten $t = 0, 1, \ldots$

$$\Delta\vec{w}_{k_l}\,(t) = \epsilon_t\,[\,\vec{x} - \vec{w}_{k_l}\,(t)] \quad \textit{bei richtiger Klassifikation} \tag{7.6}$$

und

$$\Delta\vec{w}_{k_l} = 0 \quad bei\ falscher\ Klassifikation$$

Es lernt also immer nur der Repräsentant mit dem geringsten Abstand zum Inputvektor $\vec{x} \in \Re^n$. Dieser Repräsentant kann als Sieger im Wettbewerb um die Repräsentation des aktuellen Inputs betrachtet werden. Zur "Belohnung" darf der Sieger lernen (*Wettbewerbslernen*). Ergebnis des Lernschrittes ist eine Annäherung des *best match* Repräsentanten an **einen** der zu seiner Klasse gehörigen Merkmalsvektoren. Über viele Lernschritte erfolgt somit eine langsame "Diffusion" der Repräsentanten einer bestimmten Klasse in das für diese Klasse relevante Gebiet des Datenraumes. Die Gebiete zwischen den einzelnen Klassen werden durch einen sich aus der *Voronoi-Parkettierung* ergebenden Polygonzug begrenzt.

Probleme bei diesem Lernverfahren sind sog. *dead units*, d. h. Repräsentanten $\vec{w}_{k_l}$ die auf der Trainingsmenge niemals Sieger werden und demzufolg überflüssig sind (Speicherplatz!) und die Tatsache, daß man die Zahl der Repräsentanten pro Klasse (die die Struktur der Gebietsgrenze bestimmt) von außen vorgeben muß.

Varianten des LVQ

Eine Erhöhung der Lerngeschwindigkeit und teilweise Verringerung von Fehlklassifikationen ergibt sich bei folgendem Lernalgorithmus

$$\Delta\vec{w}_{k_l} = \epsilon_t\,[\,\vec{x} - \vec{w}_{k_l}] \quad \text{bei richtiger Klassifikation}$$

und

$$\Delta\vec{w}_{k_l} = -\epsilon_t\,[\,\vec{x} - \vec{w}_{k_l}] \quad \text{bei falscher Klassifikation}$$

Ein klassenfremder Prototypvektor $\vec{w}_{k_l}$ wird also vom Datenpunkt $\vec{x}$ wegbewegt. Eine weitere Verbesserung stellt der **LVQ2** dar, bei dem der Lernschritt für den Prototypvektor $\vec{w}_{k_l}$ nur ausgeführt wird, wenn

1. Der Vektor $\vec{x}$ fehlklassifiziert wurde, d. h. $\vec{x}$ ist seinem Repräsentanten klassenfremd.
2. Der zweitnächste Prototypvektor (zweiter Sieger) zur Klasse von $\vec{x}$ gehört,
3. der Merkmalsvektor $\vec{x}$ hinreichend dicht an der Gebietsgrenze zwischen beiden Prototypvektoren liegt.

Der Algorithmus steht mit einigen Anwendungsbeispielen als *public domain* Programm zur Verfügung.

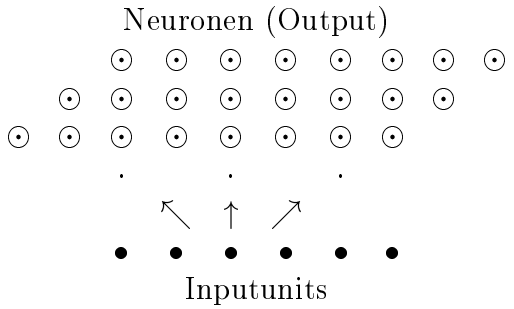
Eine Verallgemeinerung ist der dynamisch lernende Vektorquantisierer, der sich die erforderlichen Repräsentanten selbst allokiert, so daß diese nicht von Hand vorgegeben werden müssen, s. [21]. Der Algorithmus wurde u. a. erfolgreich in der Spracherkennung angewendet.

7.2 Selbstorganisation topografischer Abbildungen – der Algorithmus von Kohonen

Das reine Wettbewerbslernen sichert nur eine möglichst optimale Aufteilung des Inputraumes auf die Neuronen. Für eine innere Repräsentaton der Außenwelt muß die Abbildung allerdings auch die räumlichen Beziehungen zwischen den Reizen erhalten. Das würde eine geordnete "Verdrahtung" zwischen den Rezeptoren und den kortikalen Neuronen voraussetzen, die aber so zum Beginn der ontogentischen Entwicklung nicht vorhanden ist. Einfache informationstheoretische Überlegungen zeigen auch, daß die Information über die Organisation dieser Verschaltung nicht schon in den Genen enthalten sein kann. Im Gegenteil entsteht die geordnete Verschaltung, getrieben durch ständige Reizung der sensorischen Flächen, im Laufe der Entwicklung des ZNS quasi von allein. Für diesen Akt der Selbstorganisation gibt es eine Reihe mehr oder weniger realistischer Modelle, von denen wir hier nur den auf Kohonen zurückgehenden Lernalgorithmus für die synaptischen Gewichte besprechen wollen.

Unter einer Karte verstehen wir die **topografische** Abbildung eines Inputraumes V auf einen Outputraum $\mathcal{A}$. Solche Abbildungen werden durch spezielle einlagige neuronale Netze realisiert, d. h. der Outputraum $\mathcal{A}$ ist eine Neuronenschicht, wobei jedes Neuron von allen Inputunits direkt gespeist wird, s. Abbildung 7.1.

Die Topologie des Neuronenraumes $\mathcal{A}$ ist durch die Vorgabe der Orte $\vec{r}_i$ der Neuronen $i = 1, 2, ..., N$ festgelegt. Diese können die Knoten (Vertices) eines beliebigen Graphen sein, im konkreten Fall der Darstellung 7.7 bzw. der Abbildung 7.1 sitzen die Neuronen auf den Gitterpunkten eines zweidimensionalen Quadratgitters, d. h. $\vec{r}_i = (n_{i_1}, n_{i_2})\,,\ n_{i_j} \in \{0, 1, 2, ...\}$.



(7.7)

Jedes Neuron an einem Gitterplatz $\vec{r}$ ist durch den Vektor seiner synaptischen Gewichte $\vec{w}_{\vec{r}} \in V$ mit den Inputunits verbunden, vgl. Abb. 7.2

Da der Vektor $\vec{w}_{\vec{r}}$ in V liegt, ist er gleichzeitig die Abbildung des Gitterplatzes $\vec{r}$ nach V, s. Abb. 7.3. Die umgekehrte Abbildung $V \to \mathcal{A}$ ergibt sich durch die Vorschrift

$$\vec{x} \mapsto \vec{s} \quad \text{falls } ||\vec{w}_{\vec{s}} - \vec{x}|| \leq ||\vec{w}_{\vec{r}} - \vec{x}|| \quad \forall \vec{r}$$

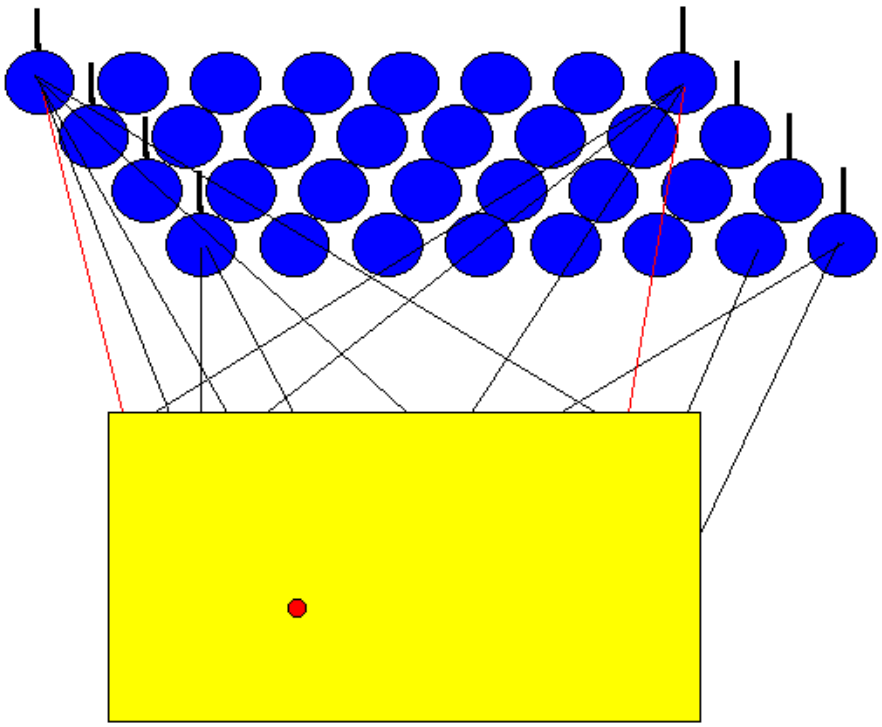


Abbildung 7.1: Abbildung einer zweidimensionalen Sinnesoberfläche (gelb) auf ein kortikales Areal, das hier durch ein zweidimensionales Gitter von Neuronen idealisiert ist. In der dargestellten postnatalen Situation sind alle Rezeptoren der Sinnesoberfläche mit allen Neuronen durch synaptische Kopplungen unterschiedlicher Stärke verbunden. Jedes Neuron sieht einen entsprechend hochdimensionalen Input. Durch Striche dargestellt sind die Kopplungen einiger weniger Neuronen mit einigen der Sinnesrezeptoren. Ontogenetisch bildet sich durch permanentes Reizen der Sinnesoberfläche eine geordnete Abbildung heraus.

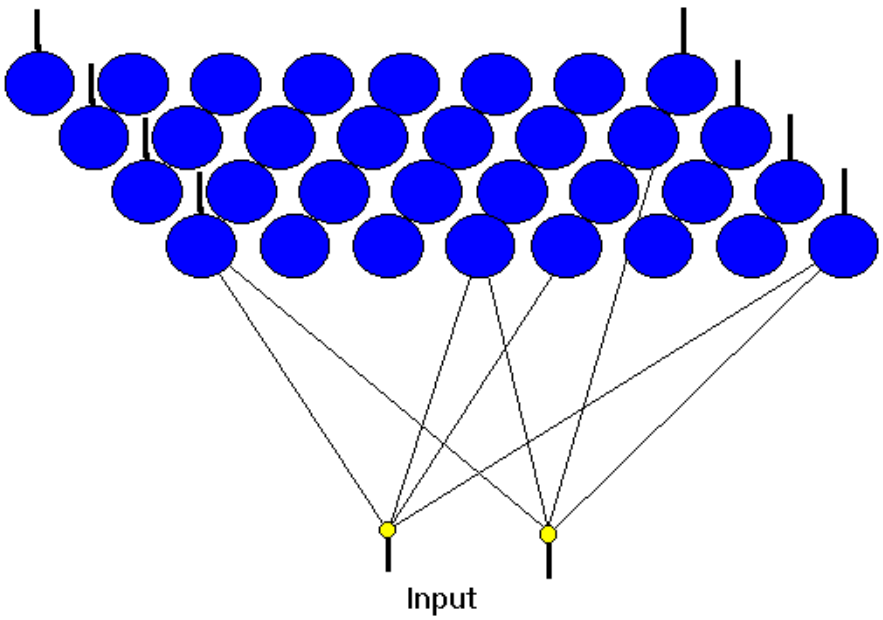


Abbildung 7.2: Für punktförmige Reize kann das in der Abbildung 7.1 dargestellte neuronale Netz vereinfacht werden. Jedes Neuron sieht nun nur zwei Inputs – die Koordinaten des Reizes auf der Sinnesoberfläche. Entsprechend ist sein synaptischer Vektor zweidimensional.

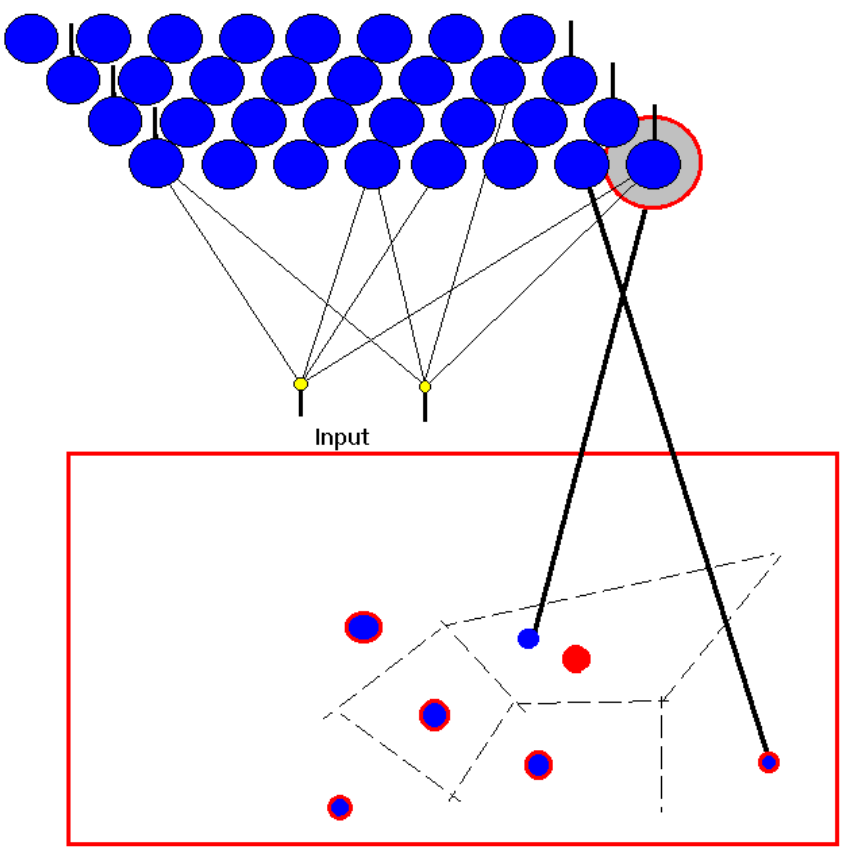


Abbildung 7.3: Für die Situation der Abbildung 7.2 sind für einige Neuronen die synaptischen Vektoren im Inputraum (blaue Punkte) zusammen mit dem Reiz (roter Punkkt) dargestellt. Die synaptischen Vektoren der Neuronen (blaue Punkte) sind die Projektionen der Neuronenorte in den Inputraum. Sie werden auch als die virtuellen Orte der Neuronen bezeichnet. Die Erregungsstärke eines Neurons bestimmt sich aus dem Euklidischen Abstand zwischen Reiz und synaptischem Vektor. Das Neuron, dessen synaptischer Vektor am besten mit dem Input übereinstimmt ist das sog. Siegerneuron (rot umrandet). Das Siegerprinzip definiert gleichzeitig eine Voronoiparkettierung des Inputraumes. Die Voronoizelle eines Neurons ist das Gebiet, dessen innere Punkte genau dieses Neuron zum Sieger machen. Die Grenzen der Voronoizellen einiger Neuronen sind gestrichelt eingezeichnet.

oder

$$\vec{s}(\vec{x}) = \arg\min_{\vec{r}} ||\vec{w}_{\vec{r}} - \vec{x}|| \tag{7.8}$$

Der Input $\vec{x}$ wird also immer auf den ihm am nächsten gelegenen synaptischen Vektor (SV) $\vec{w}_{\vec{r}}$ abgebildet. Das entspricht der Voronoi–Parkettierung des Inputraumes, s. Abb. 7.3

Um eine **topografische** Abbildung $V \to \mathcal{A}$ zu erhalten, müssen benachbarte sensorische Inputs $\vec{x} \in V$ auf die gleichen oder benachbarte Gitterplätze abgebildet werden. Mit dem von Kohonen angegebenen Lernalgorithmus für die synaptischen Kopplungen bilden sich diese Karten in einem Prozeß der Selbstorganisation heraus, daher der Name selbstorganisierende Karten (englisch self–organizing maps ⇒SOMs)[12, 13]. Der Lernalgorithmus umfaßt die folgenden Schritte:

1. Initialisierung aller synaptischen Gewichte, z. B. $\vec{w} = \vec{w}_{\vec{r}} = random\ \forall \vec{r}$. Dananch werden bei jeder Präsentation eines Inputs $\vec{x}$ folgende Schritte vollzogen:

2. **Siegerbestimmung**: Bestimmung des *best matching* SV's, d. h. finde Gitterplatz $\vec{s}$ (Neuronenort), dessen SV $\vec{w}_{\vec{s}}$ dem präsentierten Input am nächsten ist, d. h.
$$\vec{s} = \arg\min_{\vec{r}} ||\vec{w}_{\vec{r}} - \vec{x}|| \tag{7.9}$$
vgl. 7.8.

3. **Lernschritt**: Dann wird für **alle** SV $\vec{w}_{\vec{r}}$ ein Lernschritt (Adaptionsschritt) durchgeführt:
$$\Delta\vec{w}_{\vec{r}} = \epsilon h_{\vec{r},\vec{s}} \left(\vec{x} - \vec{w}_{\vec{r}}\right) \tag{7.10}$$
wobei die Nachbarschaftsfunktion
$$h_{\vec{r},\vec{s}} = \exp\left(-\frac{d\left(\vec{r},\vec{s}\right)^2}{2\sigma^2}\right) \tag{7.11}$$
die Mitlernstärke der Neuronen am Ort $\vec{r}$ bei Siegerneuron $\vec{s}$ und $d\left(\vec{r},\vec{s}\right)$ der (Euklidische) Abstand zwischen dem Siegerneuron und dem betrachteten Neuron im Gitter ist, $d\left(\vec{r},\vec{s}\right) = \sqrt{\sum_{i=1}^{2}\left(r_i - s_i\right)^2}$ für das zweidimensionale Gitter.

"Gängige Parameter": $\epsilon = 0.1$ mit langsamer Abkühlung auf Werte $\epsilon < 0.01$. Die Nachbarschaftsreichweite σ wird im Laufe des Lernens von einem anfänglich hohen Wert (so daß die Nachbarschaft etwa ein Drittel aller Neuronen umfaßt) auf einen Wert $\sigma \cong 1$ "abgekühlt". Bei der Abkühlung der Nachbarschaftsreichweite treten eine Reihe interessanter Pänomene wie z. B. Phasenübergänge auf, vgl. [16, 15] und [8, 9].

Der Ordnungsprozeß kommt durch das Mitlernen der Nachbarschaftsneuronen zustande, s. Abb. 7.4.

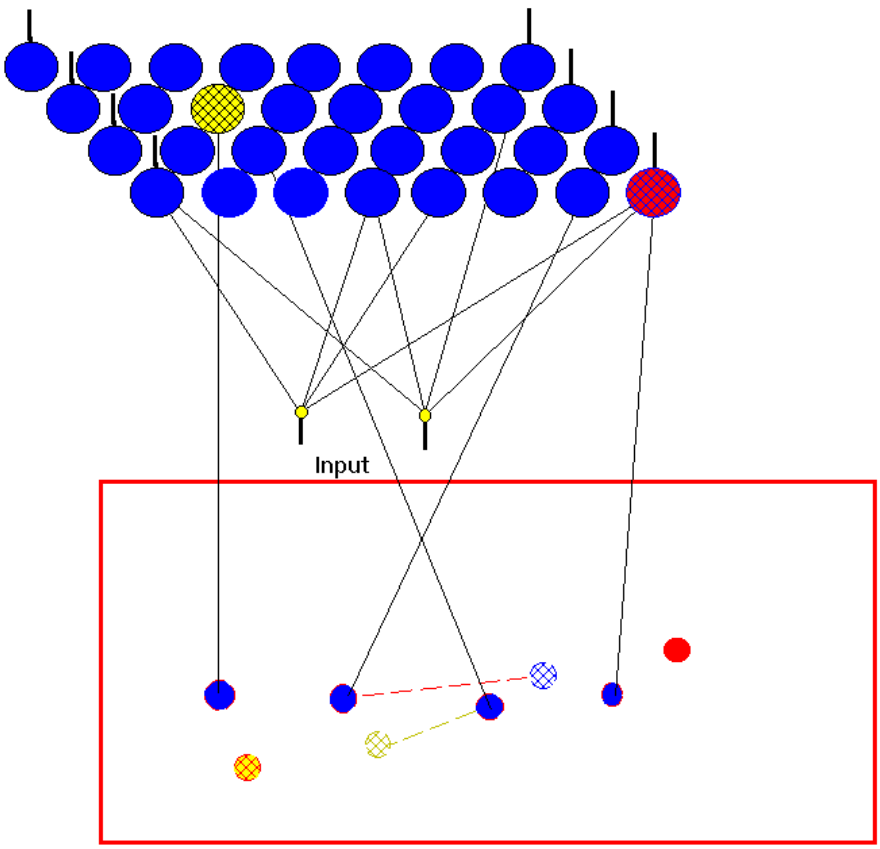


Abbildung 7.4: Der Ordnungsprozeß. Aus der Abbildung wird das selbstorganisierte Entstehen einer **topografischen** Abbildung deutlich. Dargestellt ist die Abfolge zweier Reize (rot bzw. rot schraffiert) und der Lernschritt, den ein ausgewählter Nachbar des jeweiligen Siegers vollzieht. Hierbei findet das "Entfitzen" der sich kreuzenden und damit topologieverletzenden Projektionen der beiden Nachbarneuronen statt.

Die Längenskala für die Umordnung wird dabei durch die Nachbarschaftsreichweite σ bestimmt. Für große σ wird folglich zunächst eine Grobordnung hergestellt, die sich mit der Abkühlung (Verkleinerung) von σ immer weiter verfeinert.

7.3 Anwendungen der selbstorganisierenden Karten

Die Eigenschat der SOM zur topografischen Abbildung hochdimensionaler Räume auf solche niederer Dimension wurde in einer Vielzahl von Varianten angewendet. Hier sollen nur eingige ganz wenige kurz genannt werden, eine erschöpfende Bibliographie findet sich im Internet, [1].

7.3.1 Datenentrauschung

Messdaten sind oft redundant. Das ist insbesondere eine erwünschte Eigenschaft, wenn die Daten durch Meßfehler korrumpiert, also"verrauscht" sind. Redundanz bedeutet, daß die Datenvektoren in einem Raum höherer als der problemadäquaten Dimension "leben". Die topografische Abbildung des hochdimensionalen auf den eigentlich relevanten, niedrigdimensionalen Raum ist dann geeignet, das Rauschen zu beseitigen.

Wichtig ist dabei, die richtigen Parameter für die Nachbarschaftsreichweite zu finden, s. dazu [6, 7]

7.3.2 Selbstorganisierende Karten und kognitive Prozesse – Ein Modell des '*perceptual magnet*' Effektes

Beim Erlernen von Kategorien wird in wahrnehmungspsychologischen Experimenten ein sog. '*perceptual magnet*' Effekt beobachtet: Die Grenzen zwischen den Kategorien werden wesentlich besser aufgelöst als die eher prototypischen zentralen Regionen. Das wird z. B. bei Säuglingen in der akustischen Wahrnehmung von Vokalen beobachtet. Das Kategorienlernen (Erkennung von Vokalen) kann als die Ausbildung einer Karte verstanden werden. Allerdings haben mit dem Kohonenalgorithmus gelernte Karten eine datendichteproportionale Auflösung. Das bedeutet etwa für das Vokallernen, daß die häufiger stimulierten prototypnahen Muster besser unterschieden werden können als die seltener stimulierten Grenzregionen zwischen den Kategorien.

Trotzdem können selbstorgansierende Karten als Modelle des beobachteten Verhaltens beim Kategorienlernen und damit zur Erklärung des '*perceptual magnet*' Effektes dienen, wenn der Algorithmus von Kohonenen geeignet abgeändert

wird. Wir haben in [10, 11, 3] durch eine Aufmerksamkeitssteuerung des Lernprozesses erreicht, daß sich die Auflösung der Karte einstellen läßt, so daß sie die in den speziellen psychologischen Experimenten beobachteten Verhältnisse widerspiegeln und damit den '*perceptual magnet*' Effekt erklären kann.

7.3.3 Lösung eines Optimierungsproblems – der reisende Handelsmann

Das Problem des reisenden Handelsmannes besteht darin, auf kürzstem Weg eine Rundreise durch N vorgegebe Städte zu organisieren, wobei jede Stadt genau einmal besucht werden soll. Das kann als Abbildung der räumlichen Anordnung der Städte auf eine eindimensionale Kette aus ebenfalls N Gliedern verstanden werden. Diese Abbildung ist eine approximative Lösung des Optimierungsproblemes, wenn sie nachbarschaftserhaltend ist, d. h. wenn sie kurze Abstände zwischen den Städten auf kleine Abstände auf der Kette abbildet.

Die naive Anwendung des Kohonen–Algorithmus bringt aber keine besonders guten Lösungen. Eine sehr effektive Variante ist in [4] bzw. [5] beschrieben.

7.3.4 WEBSOM – Ein automatisches Verfahren zur semantischen Clusterung von Internetdokumenten

Bei diesem Verfahren werden Internetdokumente in jeweils einen Datenvektor umgesetzt. Der Kohonen–Algorithmus realisiert, getrieben durch diese Datenvektoren die Abbildung der Dokumente auf eine zweidimensionale Fläche. Wegen der nachbarschaftserhaltenden Eigenschaft der Kartierung mit dem Kohonen–Algorithmus werden dabei benachbarte Dokumente auch auf benachbarte Regionen der Karte abgebildet. Die Software–Realisierung bietet die Möglichkeit des Zoomens, d. h. wenn man auf der Karte der Dokumente eine interessante Region gefunden hat kann man weiter in die Tiefe gehen und sich eine Karte dieses Ausschnittes auf den Bildschirm holen. Näheres und Demos unter [2].

7.4 Messung der topologischen Eigenschaften der Karten

Wie in Kapitel 7.3 dargestellt, werden die selbstorganisierenden Karten vor allem für die dimensionsreduzierende Abbildung von Datenräumen eingesetzt, zumindest läuft die Mehrzahl der Anwendungen auf dieses Problem hinaus.

Im allgemeinen ist die wahre Dimensionalität der Datenverteilung unbekannt, die Daten bilden eine nichtlineare Mannigfaltigkeit, die außerdem verrauscht ist. Geometrisch heißt das, daß die Datenvektoren im wesentlichen (d. h. bis auf das

Rauschen) auf einer gekrümmten Hyperfläche liegen, die in den hochdimensionalen Datenraum eingebettet ist.

Wenn die Dimension d der Hyperfläche bekannt und das Rauschen nicht zu stark ist, so kann die **topographische** Abbildung auf ein d–dimensionales Gitter gelingen. In der Praxis bestimmt man d durch Probieren und wegen des Rauschens kann die Topologieerhaltung nur näherungsweise gesichert werden. Damit ergibt sich die Aufgabe, die Güte der Topologieerhaltung zu messen.

Dafür sind eine Reihe von Verfahren bekannt, die aber alle schlecht für den Fall **nichtlinearer** Datenmannigfaltigkeiten taugen. Eine relativ allgemeine Lösung des Problems wurde in den Arbeiten [20, 18, 17, 19] gefunden und für eine Reihe von Anwendungsfällen getestet.

Literaturverzeichnis

[1] Bibliography on the self-organizing map (som) and learning vector quantization (lvq). http://liinwww.ira.uka.de/bibliography/Neural/SOM.LVQ.html, 1997. 2952 references up to March 1997.

[2] WEBSOM - Self-organizing map for internet exploration. http://websom.hut.fi/, 1997.

[3] H.-U. Bauer, R. Der, and M. Herrmann. Controlling the magnification factor of self-organizing feature maps. *Neural Computation*, 8(4):757–771, 1996.

[4] M. Budinich. A Self-Organizing neural network for the traveling salesman problem that is competitive with simulated annealing. *Neural Computation*, 8(2):416–424, 1996.

[5] M. Budinich and J. G. Taylor. On the ordering conditions for Self-Organizing Maps. *Neural Computation*, 7(2):284–289, 1995.

[6] R. Der, G. Balzuweit, and M. Herrmann. Building nonlinear data models with self-organizing feature maps. In *Lecture Notes in Computer Science 1112: Artificial Neural Networks*, Proc. ICANN'96 - Bochum, pages 821 – 826. Springer, 1996. Internet address: http://www.informatik.uni-leipzig.de/ der/Veroeff/bochum.ps.gz.

[7] R. Der, G. Balzuweit, and M. Herrmann. Constructing principal manifolds in sparse data sets by self-organizing maps with self-regulating neighborhood width. In *Proc. ICNN'96 Washington, IEEE*, pages 480 – 483, 1996.

[8] R. Der and M. Herrmann. Critical phenomena in self-organized feature maps: A ginzburg-landau approach. *Phys. Rev. E*, 49(5):5840–5848, 1994.

[9] R. Der and M. Herrmann. Instabilities in self-organized feature maps with short neighborhood range. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN'94)*, pages 271–276, 1994.

[10] M. Herrmann, H.-U. Bauer, and R. Der. The "perceptual magnet" effect: A model based on self-organizing feature maps. In L. S. Smith and P. J. B.

Hancock, editors, *Neural Computation and Psychology, Stirling 1994*, pages 107–116, London, UK, 1994. Springer.

[11] M. Herrmann, H.-U. Bauer, and R. Der. öptimal magnification factors in self-organizing feature maps". In *Proc. ICANN'95*, volume 1, pages 75–80, Paris, 1995. EC2 & Cie.

[12] T. Kohonen. *Self–Organization and associative memory*, volume 8 of *Springer Series in Information Science*. Springer, 1984.

[13] T. Kohonen. *The self—organizing map*. Springer, 1995.

[14] Y. Linde, A. Buzo, and M. Gray R. An algorithm for vector quantizer design. *IEEE Trans. Comm.*, 28:48–95, 1980.

[15] H. Ritter, T. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley, Reading, MA, 1992.

[16] H. J. Ritter, T. M. Martinetz, and K. J. Schulten. *Neuronale Netze: Eine Einführung in die Neuroinformatik selbstorganisierender Abbildungen*. Addison-Wesley, Munich, Germany, 1990.

[17] T. Villmann. *Topologieerhaltung in selbstorganisierenden Karten*. PhD thesis, Universität Leipzig, 1995.

[18] T. Villmann, R. Der, M. Herrmann, and T. Martinetz. Topology preservation in sofms: General definition and efficient measurement. In B. Reuch, editor, *Informatik aktuell – Fuzzy-Logic*, pages 159–166, Berlin, 1994. Springer.

[19] T. Villmann, R. Der, M. Herrmann, and T. M. Martinetz. Topology preservation in self-organizing feature maps: Exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266, Mar. 1997.

[20] T. Villmann, R. Der, and T. Martinetz. A novel approach to measure the topology preservation of feature maps. In M. Marinaro and P. G. Morasso, editors, *Proc. ICANN'94, Int. Conf. on Artificial Neural Networks*, volume I, pages 298–301, London, UK, 1994. Springer.

[21] A. Zell. *Simulation neuronaler Netze*. Addison–Wesley, 1994.