Selbstorganisierende Merkmalskarten

- **Motivation (Gehirn)**
- **Netzwerk-Architektur**
- **Topographische Merkmalskarten (Früher: „Kohonen-Karten“)**
- **Selbstorganisierende Merkmalskarte (Kohonen-Lernregel)**
- **Anwendungsbeispiele**

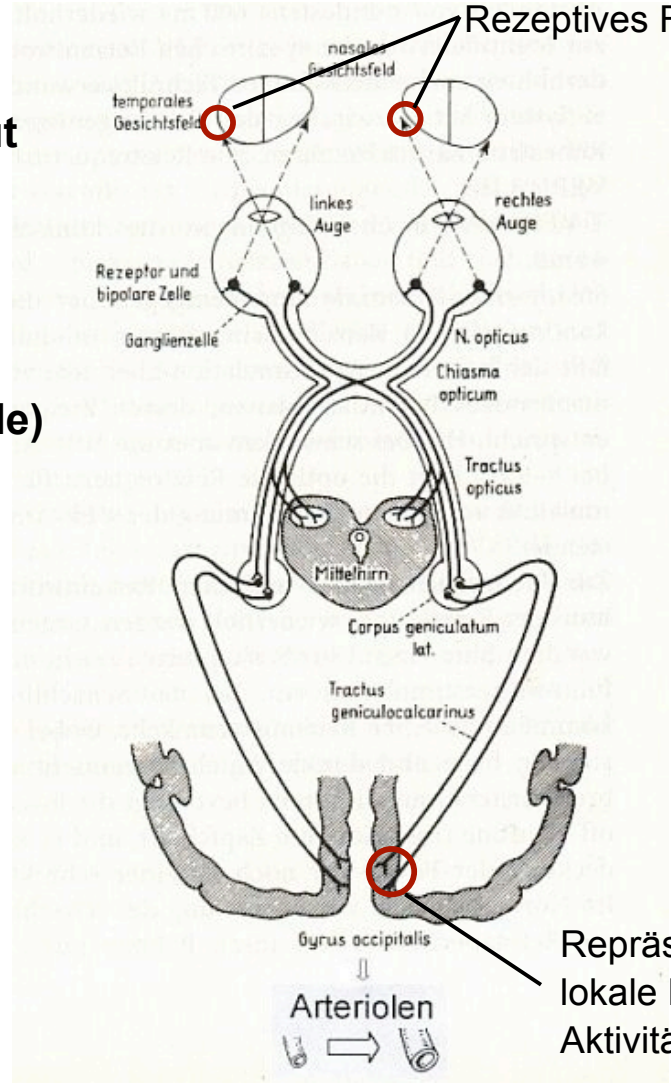
Motivation: Topografische Merkmalskarten im Gehirn

Frühe Sehbahn:

- **Zellen der primären Sehrinde verarbeiten Input aus lokalem Bereich: „Rezeptives Feld“**
- **Benachbarte rezeptive Felder erregen benachbarte Kortexbereiche**
- **Benachbarte Stimuluseigenschaften (Merkmale) erregen benachbarte Kortexbereiche**

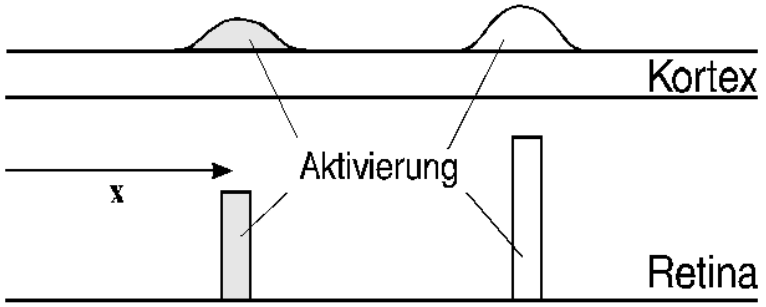
Also: Struktur interner Repräsentationen:

- **Merkmale der Umwelt werden durch den Ort der stärksten Aktivierung in der Großhirnrinde kodiert („Merkmalskarte“).**
- **Diese Kodierung ist stetig, d.h. benachbarte kortikale Orte kodieren ähnliche Reizmerkmale („Topographisch“).**



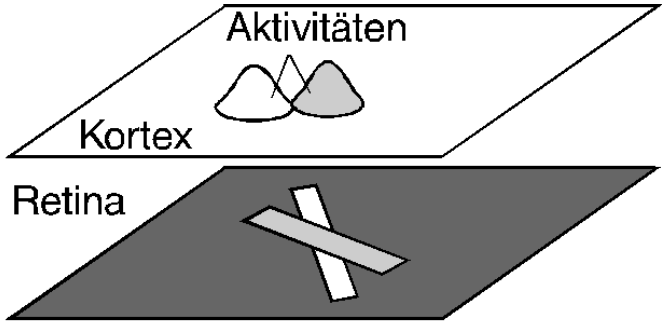
Beispiel „Retinotopie“:

- **Erscheinungsort eines Merkmals wird durch den Ort kortikaler Erregung kodiert**



Beispiel „Orientierungspräferenz-Karte“:

- **Reizorientierung wird durch den Ort kortikaler Erregung kodiert**



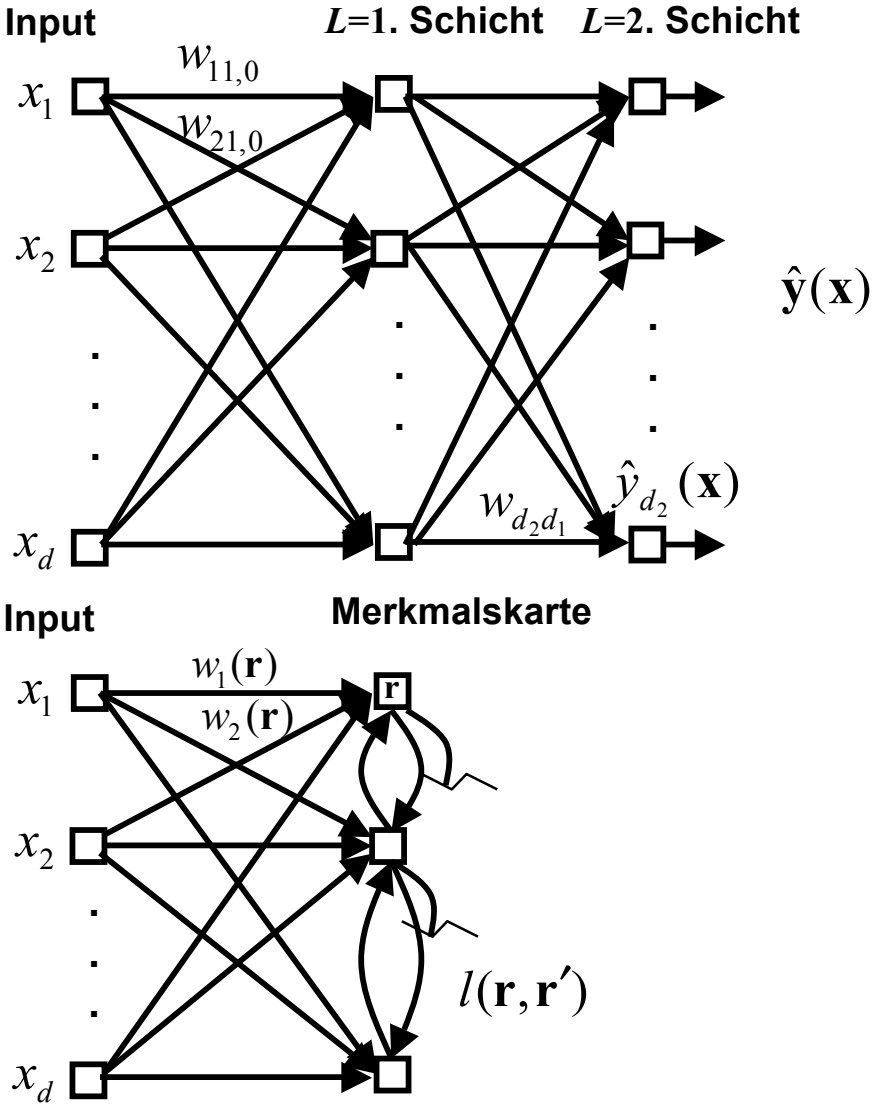
Kohonens Idee:

- **Benutze topographische Merkmalskarte als Prinzip zur Datenrepräsentation**
- **Jedes Neuron repräsentiert einen Teil des Datenraums**
- **Wo viele Daten sind, sind viele Neuronen zuständig => Dichteschätzung**
- **Gute Datenrepräsentation wird gelernt**

Implementierung in einem Neuronalen Netz

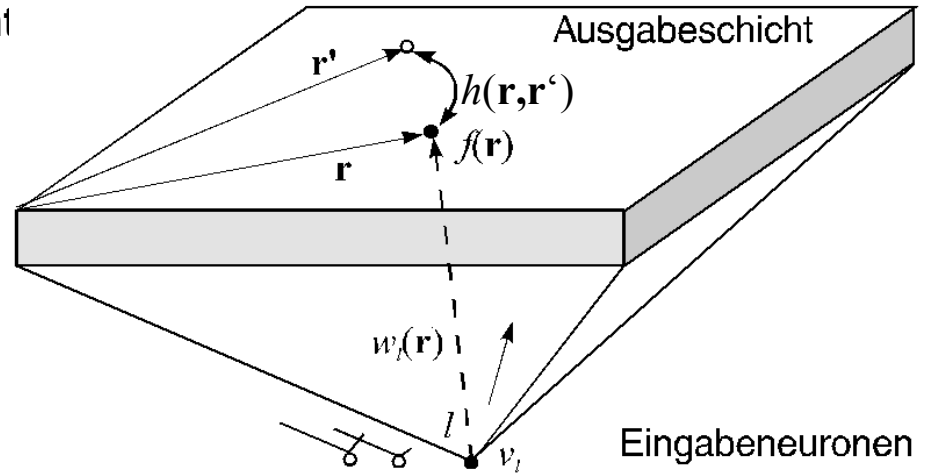
- **MLP:**
  - **Feed-Forward**
  - **Input-Neuronen verantwortlich für Halbraum**
  - **Verarbeiteter Input wird zu Output transformiert**
  - **Überwachtes Lernverfahren**

- **Selbstorganisierende Merkmalskarte:**
  - **Rückgekoppelte Verbindungen**
  - **Input-Neuronen verantwortlich für lokalisierten Bereich (wie RBF)**
  - **Verarbeiteter Input wird zu sich selbst in Verbindung gesetzt**
  - **Unüberwachtes Lernverfahren**

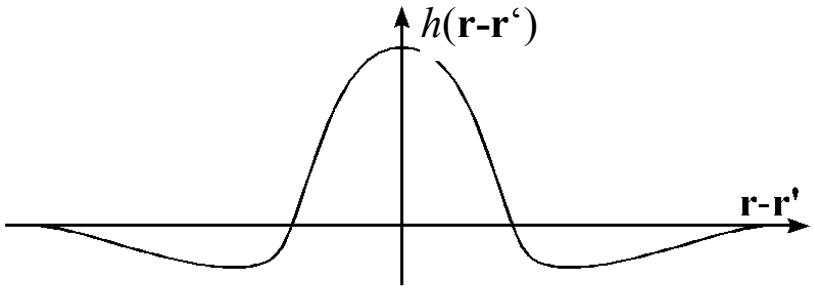


Kohonen-Netzwerk: Verschaltungsstruktur

- **Rekurrent vernetzte Ausgabeschicht**



- **Mexican-Hat-förmige, zeitunabhängige laterale Wechselwirkung**



Im Gegensatz zum Hopfieldnetz sind die lateralen Verbindungen *h*(r,r') im Ortsraum festgelegt. Biologisch motiviert: Mexican-Hat-Struktur („Umfeldhemmung“).

Netzwerk-Dynamik:

- **Die Inputneuronen initialisieren ein Aktivitätsmuster in der Ausgabeschicht**
- **Dieses wird durch die rückgekoppelte Netzwerkdynamik verändert, läuft in einen Attraktor**
- **Dynamische Gleichung:** $\frac{d}{dt} f(\mathbf{r},t) = -f(\mathbf{r},t) + g\left(\sum_{l=1}^{d} w_l(\mathbf{r}) x_l + \int d\mathbf{r}'\, h(\mathbf{r}-\mathbf{r}') f(\mathbf{r}',t) - \theta \right)$
- **Analytische Fixpunkt-Lösung schwierig, aber:**

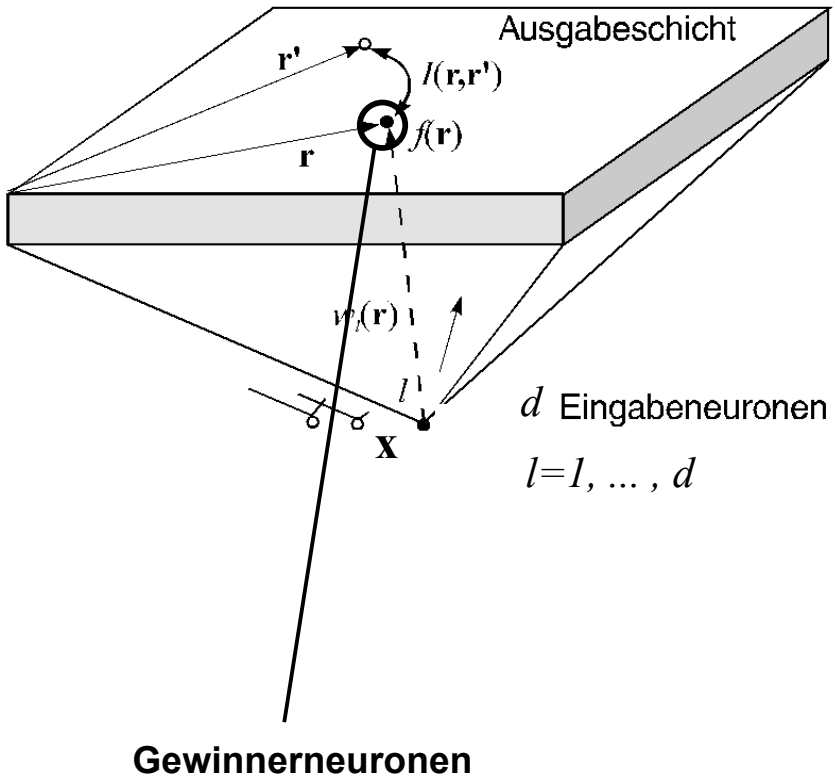
Beobachtung:

- **Laterale Wechselwirkungen vom Mexican-Hat-Typ mit genügend starker Inhibition führen immer zur Ausbildung lokalisierter Aktivitäts-"Blobs"**
- **Denn: Im Laufe der Iterationen inhibiert das anfangs am stärksten aktivierte Neuron seine Nachbarn am stärksten, vermindert damit deren inhibitorische Wirkung, kann so immer stärker aktiv werden, u.s.w...**
- **Also: Das anfangs am stärksten aktivierte Neuron + Nachbarn gewinnen:**
- **Winner-take-all Aktivierung**

Das Kohonen-Modell: „Self-Organizing Feature-Map“ (SOM)

Architektur einer Selbstorganisierenden Merkmalskarte

- ***d*** **Eingabeneuronen senden Inputs zu allen Neuronen im zweidimensionalen Gitter der Ausgabeschicht.**
- **Die Ausgabeneuronen stehen durch eine Nachbarschaftsfunktion miteinander in Beziehung**



Merkmalskarte durch „Winner-Take All“

- **Das Neuron mit dem stärksten Input sowie seine Nachbarn erhalten den „Zuschlag“, dürfen also den Input repräsentieren**

Selbstorganisation:

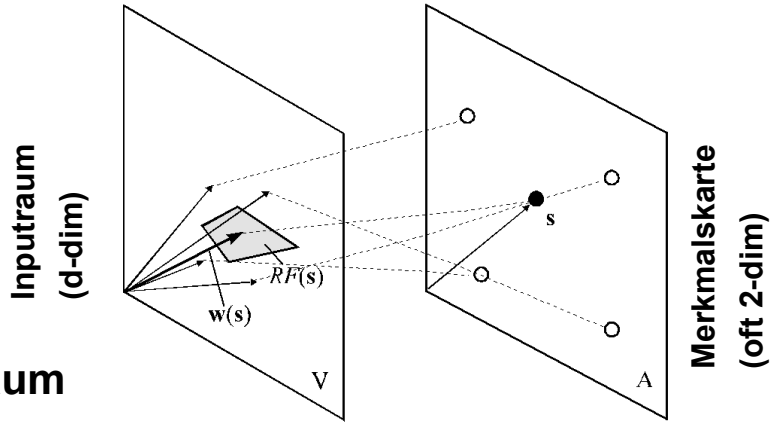
- **Gewichtsvektoren der aktivierten Nachbarn rücken näher zueinander**

Def: Merkmalskarte:

- **Abbildung, die jedem Vektor des Inputraumes (Musterraum, Merkmalsraum) einen Ort $\mathbf{s}$ in einer repräsentierenden Schicht (Karte) zuweist**

$$\phi_{\mathbf{w}} : \mathbf{x} \rightarrow \phi_{\mathbf{w}}(\mathbf{x}) = s \,\Big|\, \|\mathbf{w}(\mathbf{s}) - \mathbf{x}\| = \min_{\mathbf{r}} \|\mathbf{w}(\mathbf{r}) - \mathbf{x}\|$$

- **Bem: Die Struktur der Karte hängt von den Gewichtsvektoren ab**



Berechnung der Gewinner:

- **„Merkmal“ = Vektor $\mathbf{x}$ im d-dimensionalen Inputraum**
- **Gewichtsvektor $\mathbf{w}$ jedes Neurons lebt gleichermaßen im d-dimensionalen Inputraum**
- **Gewinner-Neuron: Neuron $\mathbf{s}$, dessen normalisierter Gewichtsvektor am nähesten am normalisierten Datenpunkt ist**

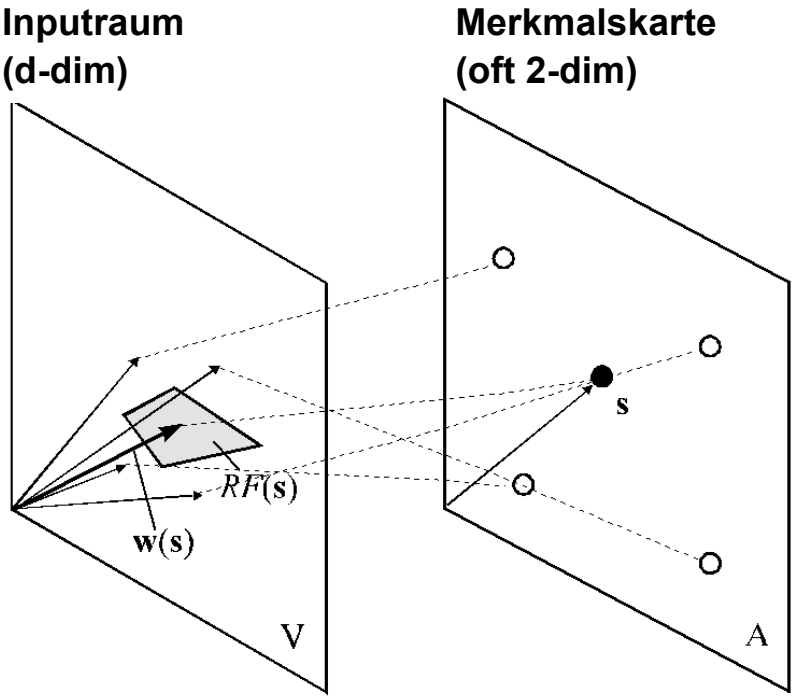
$$\mathbf{s} = \arg\min_{\mathbf{r}} \left(\|\mathbf{w}(\mathbf{r}) - \mathbf{x}\| \right)$$

- **Das Gewinner-Neuron und seine Nachbarn werden gemäß der Funktion $l(\mathbf{s},\mathbf{r})$ aktiviert: Es entsteht ein lokaler Aktivitäts-Blob um $\mathbf{s}$**
- **Bsp:** $$l(\mathbf{s},\mathbf{r}) = \exp\left(\frac{-(\mathbf{s}-\mathbf{r})^2}{2\sigma^2} \right)$$

Rezeptive Felder und Merkmalskarten :

- **Beobachtung: Jeder Gewichtsvektor / jedes Neuron ist Gewinner in einem ganzen Abschnitt des Inputraumes: „Rezeptives Feld“**

$$RF(\mathbf{s}) = \left\{ \mathbf{x} \in V \;\middle|\; \left\| \mathbf{w}(\mathbf{s}) - \mathbf{x} \right\| = \min_{\mathbf{r}} \left(\left\| \mathbf{w}(\mathbf{r}) - \mathbf{x} \right\| \right) \right\}$$



- **<u>Topographische</u> Merkmalskarte: Benachbarte rezeptive Felder sollten zu benachbarten Neuronen in der Karte gehören**
- **Ziel: Lernregel, die eigenständig diese topographische Ordnung herstellt: „Selbstorganisierende Merkmalskarte“**

Vorteile:

- **Nachbarschaftsbeziehungen im „unübersichtlichen“ Inputraum können direkt in der Ausgabeschicht abgelesen werden**
- **Auch andere Eigenschaften repräsentierbar; z.B.: Punktdichten => Dichteschätzung**

Die Kohonen-Lernregel :

- **Idee: Für jeden Datenvektor: Nähere die Gewichtsvektoren des Gewinners und seiner Nachbarn in der Karte dem Inputmuster an.**
- **Dadurch erhalten Nachbarneuronen schließlich benachbarte rezeptive Felder**
- **Dadurch werden Regionen mit vielen Datenpunkten durch viele Vektoren repräsentiert (Dichteschätzung)**

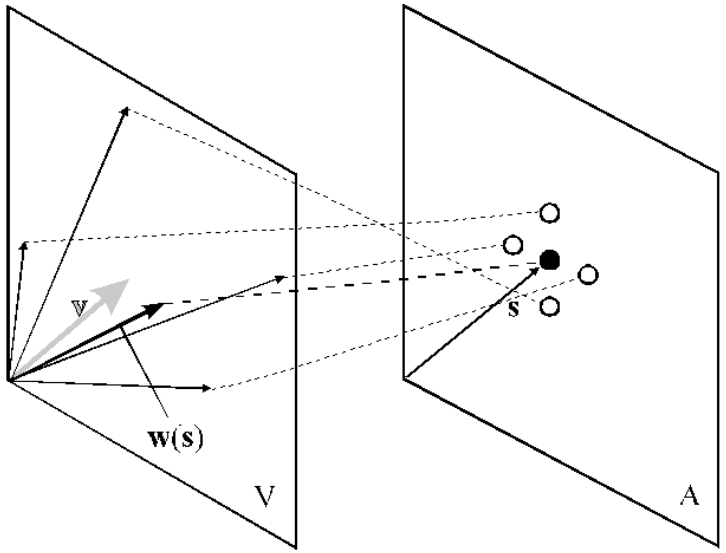
Algorithmus:

- **Geg: Datensatz auf Länge 1 normierter Datenvektoren** $D = (\mathbf{x}^{(1)},...,\mathbf{x}^{(m)},...,\mathbf{x}^{(M)})$
- **Belege Gewichtsvektoren mit Zufallswerten**
- **Präsentiere Datenvektor** $\mathbf{x}^{(m)}$
- **Ermittle den Gewinner** $\mathbf{s}^{(m)} = \mathbf{s} \left| \left\| \mathbf{w}(\mathbf{s}) - \mathbf{x}^{(m)} \right\| = \min_{\mathbf{r}} \left\| \mathbf{w}(\mathbf{r}) - \mathbf{x}^{(m)} \right\| \right.$
- **Nähere Gewichtsvektoren proportional zur Nachbarschaftsfunktion zueinander an**

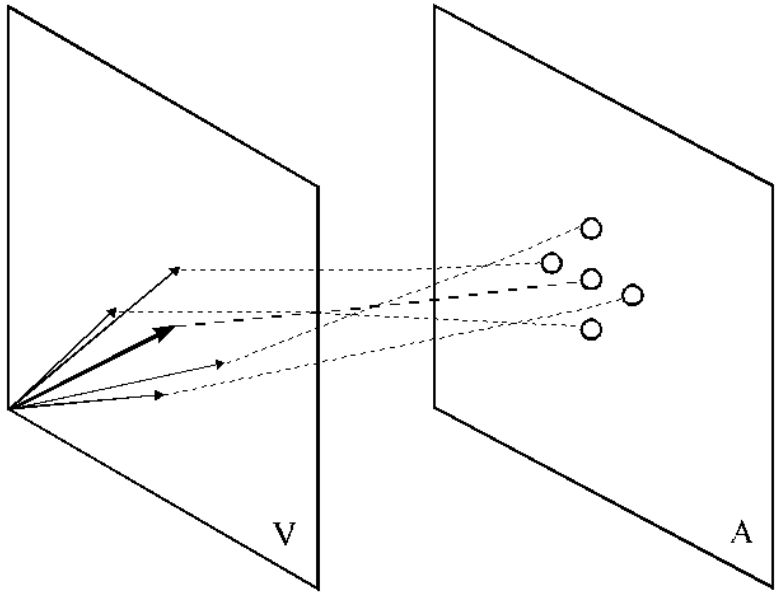
$$\mathbf{w}^{\text{neu}}(\mathbf{r}) = \mathbf{w}^{\text{alt}}(\mathbf{r}) + \Delta\mathbf{w}(\mathbf{r}) \quad \text{mit} \quad \Delta\mathbf{w}(\mathbf{r}) = \eta\, l(\mathbf{s},\mathbf{r})\left(\mathbf{x}^{(m)} - \mathbf{w}(\mathbf{r})\right)$$

- **Normiere Gewichtsvektoren auf Länge 1, präsentiere nächsten Datenpunkt**

Effekt der Lernregel:



Vor dem Training



Nach dem Training

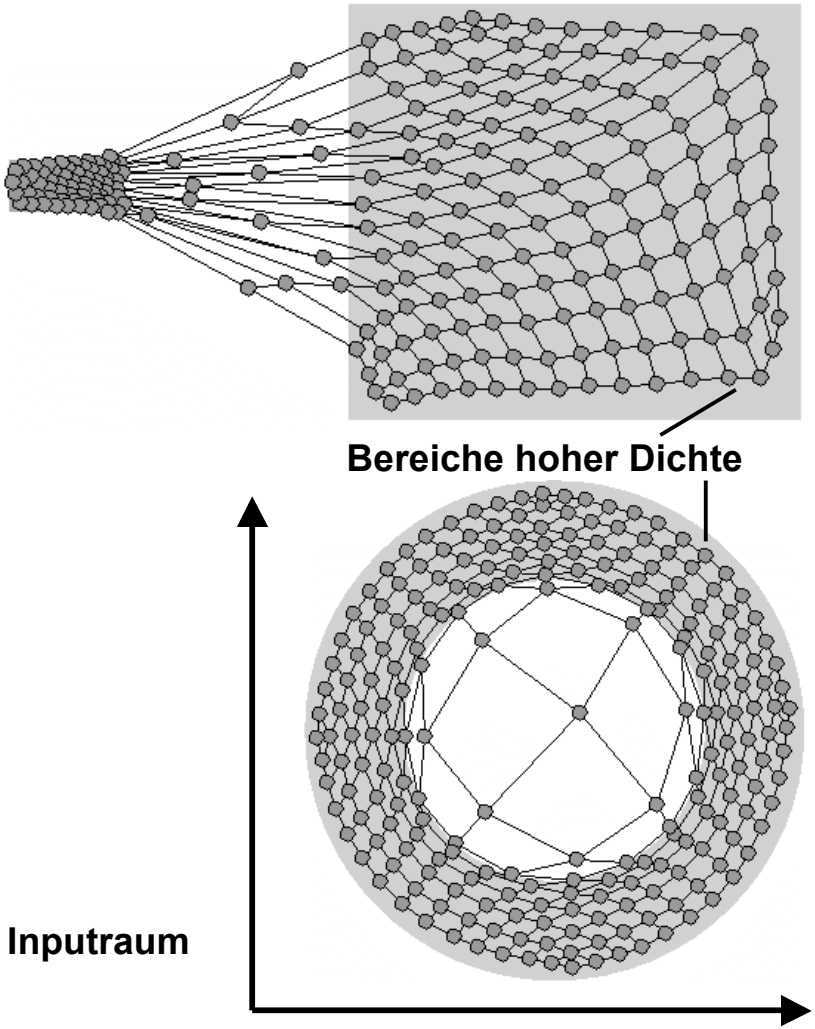
Bemerkungen:

- **Üblicherweise werden die Breite σ der Nachbarschaftsfunktion und die Lernschrittweite η im Lauf der Zeit verringert: $\sigma = \sigma(t)$, $\eta = \eta(t)$.**
- **Konvergenzbeweise gegen einen statistisch beschreibbaren Gleichgewichtszustand existieren für:** $\eta(t) = \eta\, t^{-\alpha},\ 0 < \alpha \leq 1$ $\qquad \lim_{t \to \infty} \sigma(t) = 0$

Anwendungsbeispiele

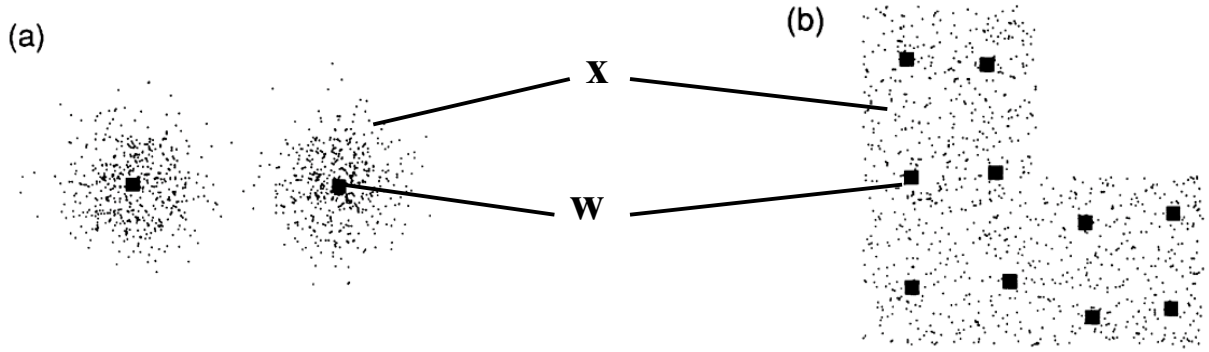
Dichteschätzung:

- **Schätzen einer Wahrscheinlichkeitsdichte, die den Daten zugrunde liegt**
- **Die Merkmalskarte wird mit Vektoren trainiert, die aus der zu schätzenden pdf als Stichproben gezogen wurden**
- **Häufig auftretende Merkmale werden von der SOM durch mehr Neuronen repräsentiert**
- **Beispiel: Merkmalskarte mit 15x15 Neuronen**



Spezialfall Vektorquantisierung: **(Verfahren zur Datenkompression)**

- **Folge von Datenvektoren $\mathbf{x}(t)$, $t = 1, \ldots,$ sollen durch eine feste Anzahl von Referenzvektoren $\mathbf{w}_s$ approximiert werden**
  - **Kompression: Speicherung des Index $s(\mathbf{x})$ mit minimalem $\|\mathbf{w}_{s(\mathbf{x})} - \mathbf{x}\|$ für jedes $\mathbf{x}(t)$**
  - **Restauration: $\mathbf{x}(t) := \mathbf{w}_{s(\mathbf{x}(t))}$, $t = 1, \ldots$ (Es gibt einen Restaurationsfehler!)**
- **Ziel: Finde optimale Verteilung der Referenzvektoren mit min. Restaurationsfehler**
- **Kostenfunktion** $$E = \int P(\mathbf{x}) \left\| w_{s(\mathbf{x})} - \mathbf{x} \right\|^2 d\mathbf{x} \equiv \min$$
- **Gradientenabstieg** $$\mathbf{w}_s(t+1) = \mathbf{w}_s(t) - \frac{\eta}{2} \frac{\partial E}{\partial \mathbf{w}_s} = \mathbf{w}_s(t) + \eta \int_{RF(s)} \left(\mathbf{x} - \mathbf{w}_s(t)\right) P(\mathbf{x})\, d\mathbf{x}$$
- **Empirische Lernregel** $$\mathbf{w}_{s(\mathbf{x})}(t+1) = \mathbf{w}_{s(\mathbf{x})}(t) + \eta\left(\mathbf{x} - \mathbf{w}_{s(\mathbf{x})}\right)$$
- **Bsp:**

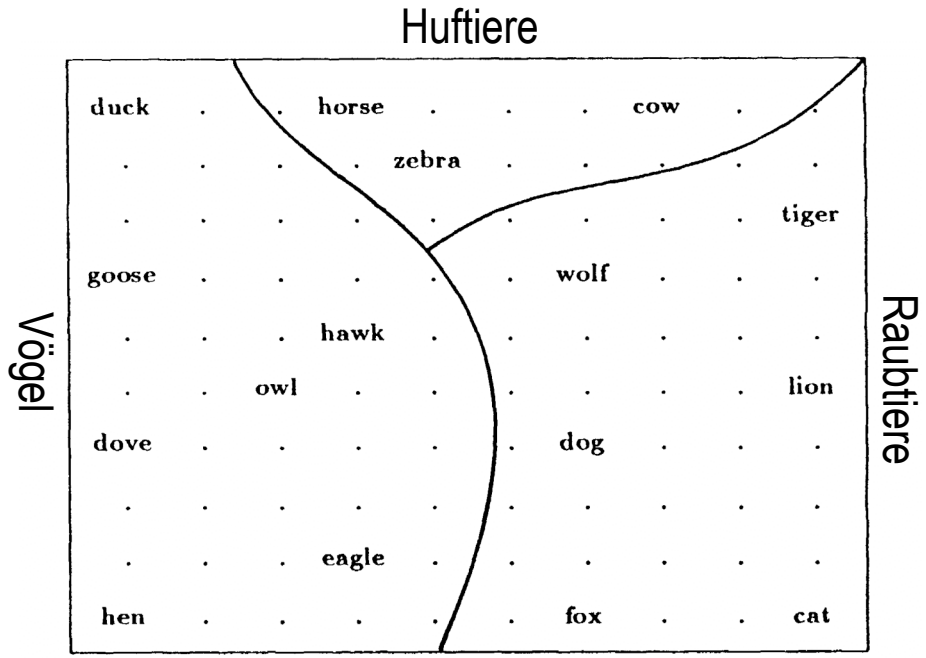


Clustering und Visualisierung:

- **Problem: Hochdimensionaler Datenraum**
- **Aufgabe: Finde Korrelationen in den Daten**
- **Lösung mit dem Kohonenalgorithmus: Merkmale, die im Inputraum nahe beieinanderliegen, werden auf benachbarte Gebiete der SOM abgebildet**

	klein	mittel	groß	2 Beine	4 Beine	Haare	Hufe	Mähne	Federn	jagt	rennt	fliegt	schwimmt
Taube	1	0	0	1	0	0	0	0	1	0	0	1	0
Henne	1	0	0	1	0	0	0	0	1	0	0	0	0
Ente	1	0	0	1	0	0	0	0	1	0	0	0	1
Gans	1	0	0	1	0	0	0	0	1	0	0	1	1
Eule	1	0	0	1	0	0	0	0	1	1	0	1	0
Falke	1	0	0	1	0	0	0	0	1	1	0	1	0
Adler	0	1	0	1	0	0	0	0	1	1	0	1	0
Fuchs	0	1	0	0	1	1	0	0	0	1	0	0	0
Hund	0	1	0	0	1	1	0	0	0	0	1	0	0
Wolf	0	1	0	0	1	1	0	1	0	1	1	0	0
Katze	1	0	1	0	1	1	0	0	0	1	0	0	0
Tiger	0	0	1	0	1	1	0	0	0	1	1	0	0
Löwe	0	0	1	0	1	1	0	1	0	1	1	0	0
Pferd	0	0	1	0	1	1	1	1	0	0	1	0	0
Zebra	0	0	1	0	1	1	1	1	0	0	1	0	0
Kuh	0	0	1	0	1	1	1	0	0	0	0	0	0

- **Die Merkmale werden ihrer Ähnlichkeit entsprechend auf der SOM angeordnet**
- **Topologieerhaltende Abbildung des hochdimensionalen Inputraumes auf die zweidimensionale Kartenfläche**



Spezialfall: Dimensionsreduktion

- **Geg: Signale mit vielen Freiheitsgraden (hohe Dimensionalität).**
- **Ges: Bestmögliche Repräsentation in einer niedrigdimensionalen Neuronenstruktur (typisch: 1 bis 2-dimensional).**

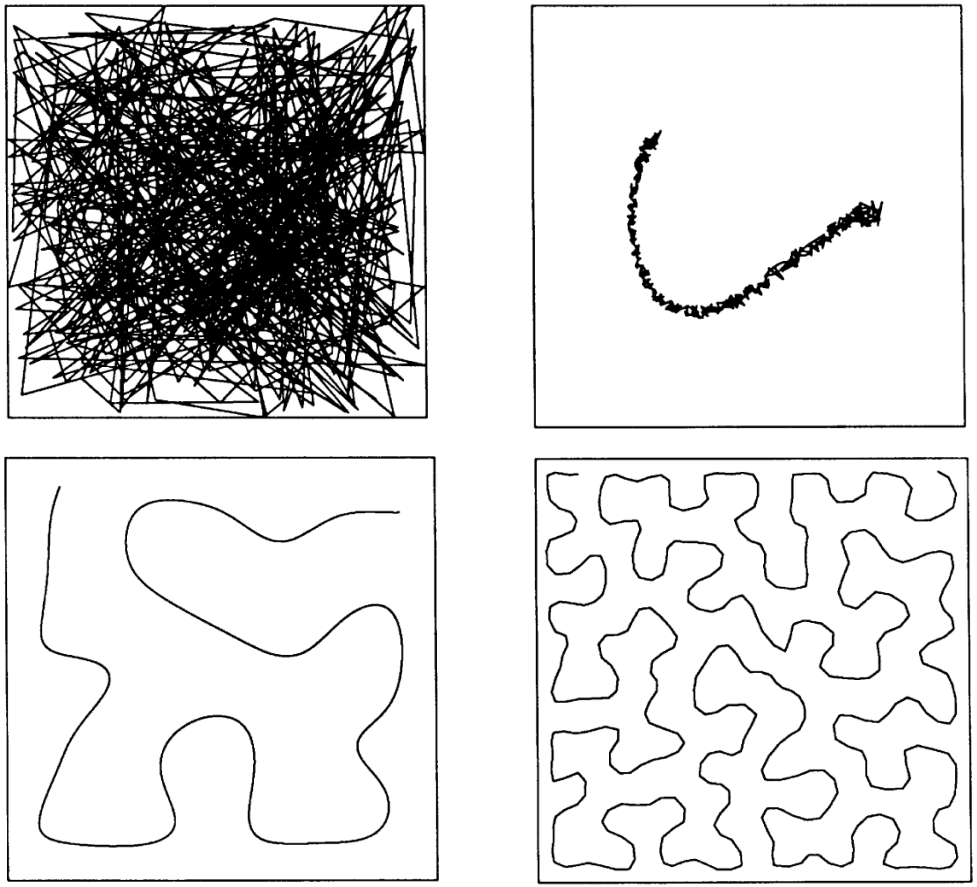
- **Der Kohonenalgorithmus führt zu einer optimalen Abdeckung des höherdimensionalen Raumes gemäß der Wahrscheinlichkeitsverteilung der präsentierten Inputmuster**

- **Beispiel: Eindimensionale Neuronenkette wird mit zweidimensionalen Vektoren aus dem Einheitsquadrat trainiert**

- **Zeitabhängige Breite der Nachbarschaftsfunktion:** $\sigma(t) = 100\left(0.01\right)^{10^{-5}t}$

- **bewirkt sukzessive Ausbildung immer feinerer Strukturen**

Dimensionsreduktion: Lernverlauf



Zuordnung: zu Beginn, nach 200, nach 50000, nach 100000 Schritten

Beispiel Sensorik: Positionskodierung einer Schallquelle

Ausgangspunkt:

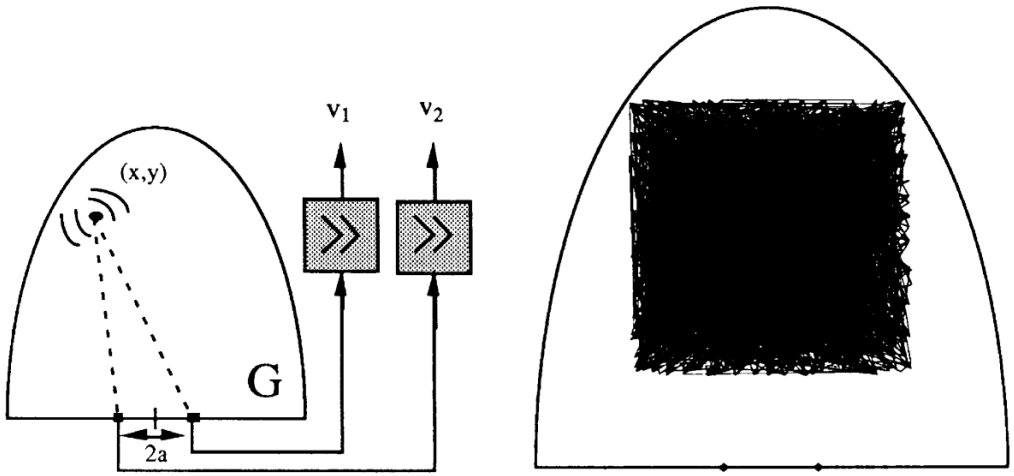
- **Schallsignale einheitlicher Lautstärke ertönen in beliebigen Positionen innerhalb eines krummlinig begrenzten Gebietes**
- **Die Signale werden von zwei Mikrofonen aufgenommen, deren Ausgangsintensität den Abstand der Schallquelle kodiert**
- **Mit der nichtlinearen Verstärkerkennlinie $f(x)$ werden die Signale zu:**

$$v_1 = f\left(\left[(x-a)^2 + y^2\right]^{-1}\right)$$

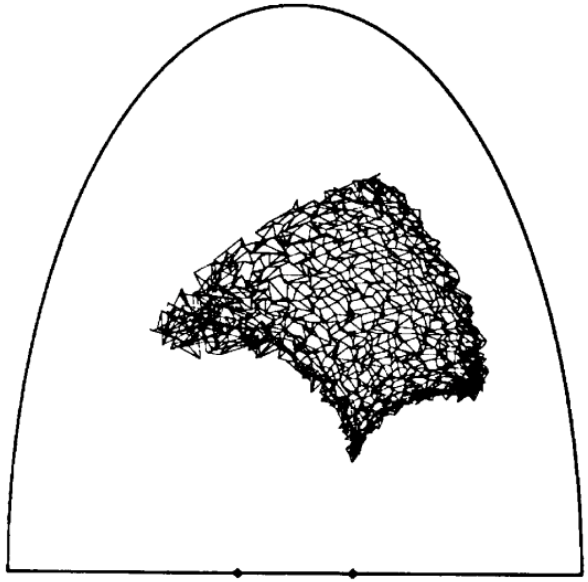
$$v_2 = f\left(\left[(x+a)^2 + y^2\right]^{-1}\right)$$

Ziel:

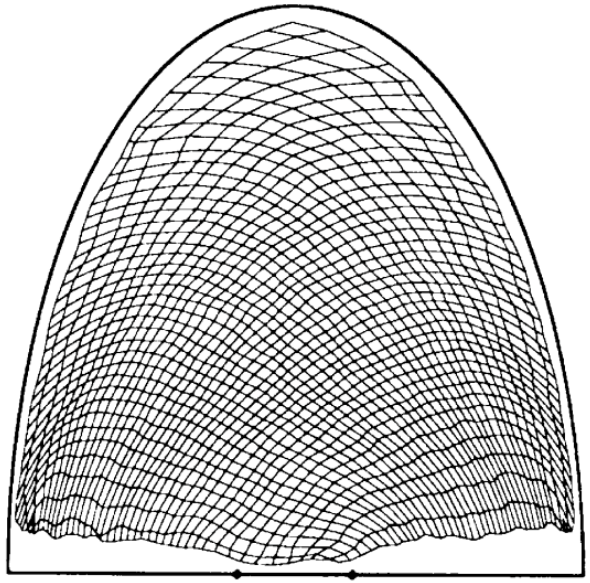
- **Lerne Merkmalskarte zur Rekonstruktion des Ortes aus dem gemessenen Schallsignal**



Positionskodierung einer Schallquelle: Lernverlauf



Karte nach 100 Lernschritten

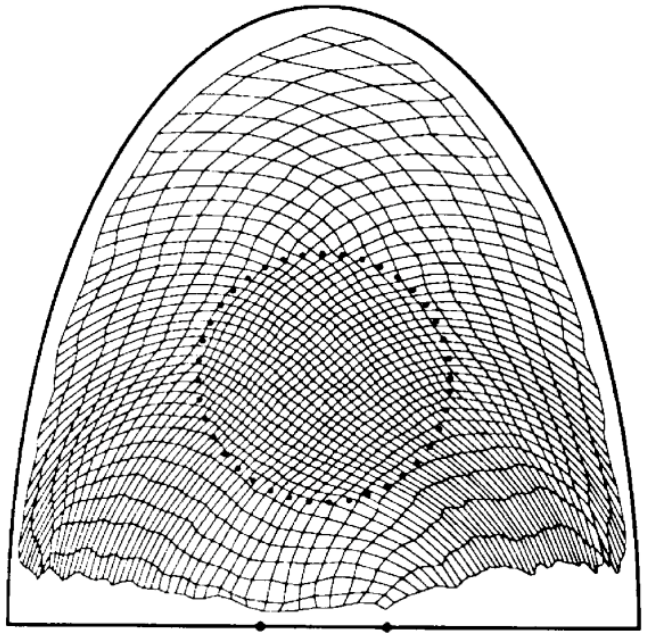


Karte nach 40000 Lernschritten

- **Es sind die Gewichtsvektoren und damit die Positionen höchster Sensitivität für ein Gitter von 40x40 Neuronen dargestellt. Nach dem Training kodiert jedes Neuron einen Teil-bereich des betrachteten Gebietes. Das Netzwerk hat die der Positionskodierung innewohnende nichtlineare Transformation invertiert.**

Positionskodierung einer Schallquelle: Dichteschätzung des sensorischen Inputs

- **Die Repräsentation passt sich der Wahrscheinlichkeitsverteilung der Inputsignale an, d.h. häufig präsentierte Muster werden durch mehr Kohonen-Neuronen kodiert**



Im zentralen Kreis wurde die Signalhäufigkeit gegenüber außen um einen Faktor 3 erhöht.

Jedes Muster hat eine „anziehende“ Wirkung auf die im Inputraum benachbarten Gewichtsvektoren: Kumulation bei Peaks der Wahrscheinlichkeitsverteilung.

(s. a. Vektorquantisierung).

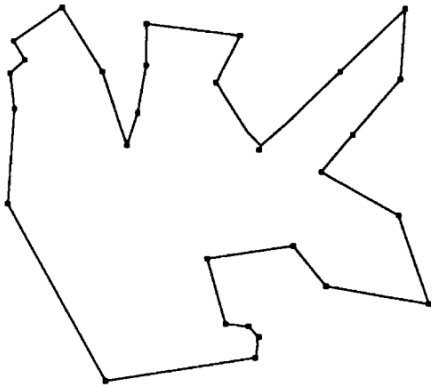
Denselben Effekt kann man durch lokale Erhöhung der Netzwerkelastizität (Verbreiterung von l) erreichen

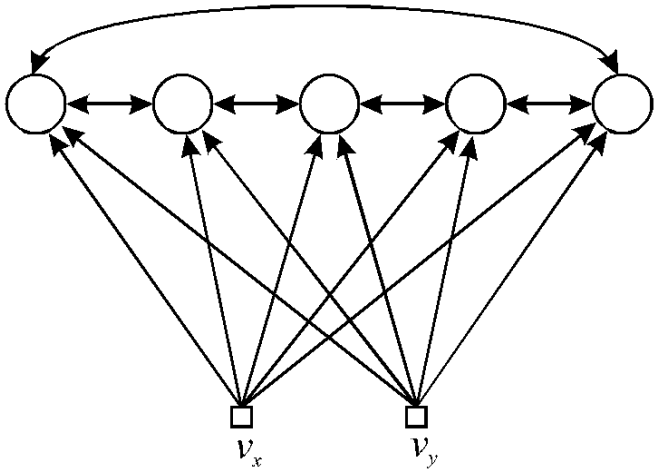
Optimierungsprobleme:

- **Lösungsaufwand für ein System mit *L* Komponenten steigt wie exp(*L*) bzw. *L!***

- **Beispiel: Handlungsreisenden-Problem:**
  - **Finde die kürzeste Route, die *L* gegebene Städte berührt.**
  - **Ansatz: Wähle eindimensionalen Neuronenring mit *N* ≥ *L* Neuronen (*N* > *L* empfohlen) und zweidimensionalen Gewichtsvektoren.**

    Die Inputsignale kodieren *x*- und *y*-Positionen der Städte.





Netzwerkarchitektur für das Handlungsreisenden-Problem. Die Inputs kodieren die xy-Position, die Ausgabeneuronen implementieren eine Ringnachbarschaft.

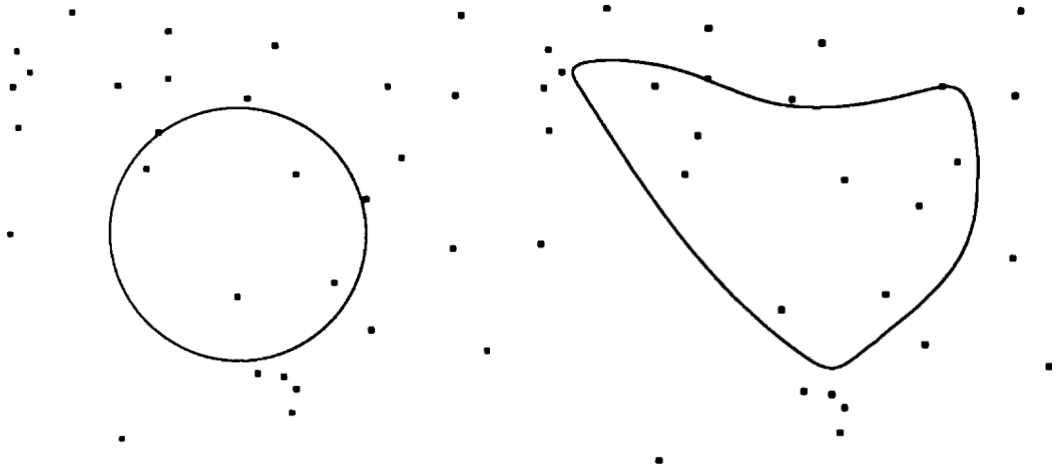
- **Präsentiere die Städtepositionen als Inputmuster und trainiere mit**

$$\Delta \mathbf{w}(r,t) = \eta(t)\, l\big(s(\mathbf{x}),r,t\big)\big(\mathbf{x} - \mathbf{w}(r,t)\big)$$

- **Die Nachbarschaftsfunktion $l(r,r')$ versucht, die Repräsentation des Rings im Ortsraum möglichst kurz zu halten („kurzer Weg“-Forderung)**
- **Unter dieser Bedingung werden die Städtepositionen sukzessive approximiert**

Simulationsablauf (nach Durbin und Willshaw 1987):

$L = 30$
$N = 800$
$h = 0.8$
$s(t) = 50\ 0.02^{(t/t\max)}$
$t_{max} = 10000$



Verlauf einer Simulation mit 30 Städten es werden die Gewichtsvektoren im zweidimensionalen Inputraum gezeigt nach 0, 5000, 7000 und 10000 Lernschritten