Kohonen's selbstorganisierte Karte

Marc Seemann
Marcus Ritt
Heike Speckmann

Institut für technische Informatik Tübingen

24/25 September 1992

Zusammenfassung

Die Änderungen, die von uns durchgeführt wurden beschränken sich allesamt auf Kohonens selbstorganisierende Karte.
Es wurde ein Tool entworfen, welches schon angelernte Karten in ein .net File konvertiert um sie im SNNS zu visualisieren.
Daneben wurden Analysewerkzeuge in SNNS implementiert, die eine Auswertung des Lernergebnisses ermöglichen. Es handelt sich zum einen um die Darstellung von Komponentenkarten und zum anderen um eine Möglichkeit die Lage von Vektoren auf der Karte zu lokalisieren.
Zu guter Letzt ist es bei Visulisierung und Analyse nicht geblieben. Wir haben den Kohonen-Algorithmus, trotz einiger Einschränkungen, die auf SNNS-interne Strukturen zurückzuführen sind, als weiteres Lernverfahren in den Simulator integriert.

1 Problembeschreibung

Die Problemstellung und ihre Umsetzung soll anschaulich am Beispiel der Produktionssteuerung bei der Herstellung von Mikrochips dargestellt werden.
Bei der Produktion von Mikrochips treten extrem lange Produktionslaufzeiten auf. Wie z.B. beim 4 Mbit Chip mit 3 Monaten. Dabei ist es wichtig, Testergebnisse so früh wie möglich zu erfassen und auszuwerten, um nötige Veränderungen an der Produktionslinie vornehmen zu können.

Datenstruktur
Die Gesamtheit aller Einflussgrössen bei den Produktionsschritten wird durch den Vektor P zusammengefasst. Die zu messenden elektrischen Eigenschaften bilden den Vektor T, den Testparameter. Vektor X stellt die Vereinigung beider Vektoren im n-dimensionalen Raum dar.

Verfahren
Eine effektive Speicherung der Vektoren ist nur unter Berücksichtigung der inneren Zusammenhänge der Häufigkeitsverteilung möglich. Dabei ist eine Abbildung des n-dimensionalen Raum auf eine 2-dimensionale Ebene wünschenswert, wobei natürlich die Korrelationen der einzelnen Komponenten nicht verloren gehen dürfen. Die Position, wo ein Eingabevektor gespeichert werden soll, wird durch einen selbstorganisierenden Prozess festgelegt. Die Auswertung erfolgt durch eine geeignete graphische Darstellung. Diese Darstellung sollte in möglichst komprimierter Weise die gespeicherten Daten wiedergeben und dabei die Zusammenhänge zwischen Prozess- und Testparametern aufzeigen. Aufgrund dieser Ergebnisse kann man dann daran gehen, die Effektivität der Produktionslinie zu erhöhen. Es besteht damit bereits im frühen Stadium die Möglichkeit festzustellen, ob die Spezifikationen des Chips erfüllt werden, um gegebenenfalls in die Steuerung des Produktionsprozesses einzugreifen.

2 Topologie der selbstorganisierenden Karte

Die selbstorganisierende Karte beruht auf dem Algorithmus von Kohonen, und wie aus dem Name bereits zu entnehmen ist, handelt es sich hierbei um ein unüberwachtes Lernverfahren. Die Karte besteht aus einer regelmässigen, matrixförmigen Anordnung von gleichartigen PEs (hidden units), wobei jedes PE mit seinem Nachbar-PE gekoppelt ist (Die zu anfangs festgelegte Nachbarschaftsbeziehung kann später in der grafischen Oberfläche von SNNS nicht mehr verändert werden!). Das PE_j speichert dabei genau einen reellen Vektor M_j, vermittels den Gewichten $(\mu_{1j}, \mu_{2j}, \ldots, \mu_{nj})$. Der am Eingang (input units) anliegende Eingabevektor X wird an die Eingänge aller PEs weitergeleitet und dort in Abhängigkeit der stochastischen Erregerfunktion $e_j(t)$ gespeichert. $e_j(t)$ ist das Maß für die Bereitschaft des j.ten PEs sich zum Zeitpunkt t an den Eingangsvektor X anzupassen.

3 Der Kohonen-Algorithmus

Im folgenden soll nun betrachtet werden, wie der Kohonen Algorithmus arbeitet. Da es eine Fülle unterschiedlicher Methoden gibt, soll die in SNNS im-

plementierte, hier kurz beschrieben werden. Dabei stellt der von uns gewählte Weg nicht unbedingt einen Weg dar, der sich anderen gegenüber besonders auszeichnet. Andererseits war es aber auch nicht möglich, sämtliche Derivate zu berücksichtigen.

Initialisierung
Die Gewichte der Karte werden zufällig initialisiert und sollten wie bei Counterpropagation auf die Länge 1 normiert sein. Nachdem wir einen Fehler aus einer bereits vorhanden Initialisierungsfunktion behoben haben, kann die Funktion *INIT_Weights_CPN* für diesen Zweck gut verwendet werden. Es ist aber auch möglich, die Karte bei der Generierung mit `convert2snns` mit eigenen Werten zu belegen.

Stimulation
Aus dem Satz der Mustervektoren wählt man einen Vektor X' aus. Bei der Übernahme an die Input Units wird der Vektor auf die Länge 1 normiert:

$$X = \frac{X'}{\|X'\|}$$

Dies geschieht mit der Funktion *normalize_inputvector()*.

Ähnlichkeitsbestimmung
Nun ermittelt man die Ähnlichkeitsbeziehung A_j des Eingabevektors X zu den gespeicherten Vektoren M_j auf der Karte. Als Ähnlichkeitsmaß dient das Skalarprodukt:

$$A_j(t) = X \cdot M_j = \sum_{i=1}^{n} \xi_i(t)\mu_{ij}(t)$$

Bei konsequenter Normierung der Vektoren, entspricht diese Form der Ähnlichkeitsbeziehung dem euklidschen Abstand der Vektoren X und M_j , da gilt:

$$X \cdot M_j = \cos(\varphi)\,\|X\|\,\|M_j\| = \cos(\varphi)$$

Der ähnlichste Vektor M_c, der durch das PE_c gespeichert ist, hat mit X das grösste Skalarprodukt bzw. den kleinsten euklidschen Abstand, es gilt also:

$$A_c(t) = \max\{A_j(t)\} = X\dot{M}_c$$

Adaption
Bezeichnet man mit N_c die Indexmenge der PEs, die sich in einer Umgebung mit Radius $w(t)$ des Maximums c befinden, so werden alle M_j mit $j \in N_c$ in Richtung des Eingabevektors X adaptiert.

$$\begin{aligned}
\frac{d\,\mu_{ij}(t)}{d\,t} &= e_j(t)\,(\xi_i(t) - \mu_{ij}(t)) && \text{für } j \in N_c \\
\frac{d\,\mu_{ij}(t)}{d\,t} &= 0 && \text{für } j \notin N_c \\
\text{mit} \quad e_j(t) &= h(t)\,e^{-\frac{(x-cx)^2+(y-cy)^2}{w^2(t)}} \\
&= h(t)\,e^{-(d_j/w(t))^2} && \text{(Gauß-Funktion)}
\end{aligned}$$

(x, y)	(Spalte,Zeile) des j.ten PEs
(cx, cy)	(Spalte,Zeile) des Maximums c
d_j	Abstand des j.ten PEs zum Maximum c
$h(t)$	Adaptionshöhe zum Zeitpunkt t ($0 \leq h(t) \leq 1$)
$w(t)$	Radius um PE_c zum Zeitpunkt t

Nach der Adaption liegen die Vektoren nicht mehr auf der Hyperkugel mit $r = 1$, daher müssen sie erneut normiert werden.

$$M_j(t_{neu}) = \frac{M_j(t)}{\|M_j(t)\|} \qquad \text{für } j \in N_c$$

Dies geschieht mit der Funktion *normalize_weight()*.

Änderung der Parameter
Wurden einmal alle Mustervektoren angelernt, so werden die Parameter Lernhöhe $h(t)$ und Einzugsradius $w(t)$ verringert. Die Änderung erfolgt durch Multiplikation mit dem Parameter *mul*, wobei $0 \leq mul \leq 1$ gelten muß:

$$h(t) := h(t)\,mul \quad ; \quad w(t) := w(t)\,mul$$

4 Arbeiten mit Kohonen und dem SNNS

convert2snns
ist ein Tool ausserhalb von SNNS, das :

- die Kohonenkarte samt Inputvektor generiert
- Gewichtsfiles im ASCII-Format in ein .net File von SNNS konvertiert
- Musterfiles im ASCII-Format in ein .pat File von SNNS konvertiert

Aufruf: `convert2snns` ⟨`Kontrolldatei`⟩

Die Kontrolldatei hat folgende Struktur:

```
Name der Musterdatei
Name der Gewichtsdatei
Grösse der Karte X
Grösse dar Karte Y
Zahl der Komponenten n
Zahl der Muster
```

Ist die Datei mit der Belegung der Karte nicht vorhanden, so wird das Netz mit Grösse $X \cdot Y$ und Komponentenzahl n generiert. Die Gewichte werden auf Null gesetzt.
Neben dem Eingabevektor und der eigentlichen Kohonenkarte ($X \times Y$ Matrix) wird ein weiteres Output-Unit erzeugt, da SNNS ohne dieses nicht arbeiten kann. Dieses ist ein Dummy und wird für Kohonen nicht benutzt, da sich der Kohonenalgorithmus auf ein 2-stufiges Netz beschränkt.
Beim Anlegen einer Kohonenkarte mit dem bignet Tool aus SNNS gibt es zur Zeit noch Kompatibilitätsschwierigkeiten.

Aufruf des Kohonen-Lernalgorithmus
Nachdem man eine Kohonenkarte mit `convert2snns` generiert hat, wählt man im Remote Panel Kohonen als Lernfunktion und muss folgende Lernparameter angeben:

1. Anfangshöhe $h(t)$
2. Anfangsradius $w(t)$
3. Multiplikative Abnahme mul
4. Horizontale Grösse X

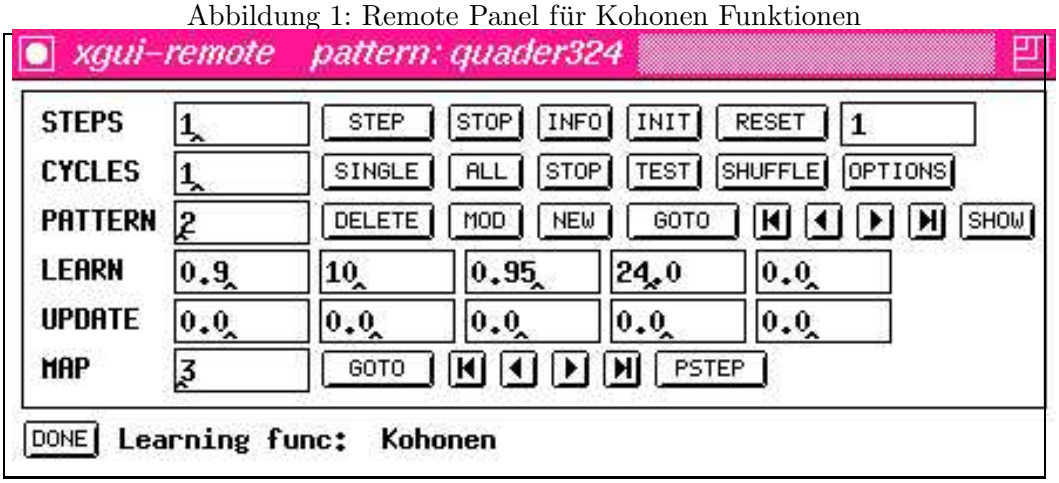
Die Karte wird mit *CPN_Weights* initialisiert.
Muster können beispielsweise mit `convert2snns` von einem ASCII-File in SNNS importiert werden. Wenn nicht, sind diese zuvor von Hand zu erzeugen. In das CYCLE Feld gibt man die Zahl der Durchgänge an und startet das Lernverfahren. Da es sich bei Kohonen um ein unüberwachtes Verfahren handelt, ist die Fehlerausgabe immer Null.

Komponentenkarten
Das Lernverfahren von Kohonen führt dazu, dass sich die Karte organisiert. Man erkennt dies daran, dass sich Cluster bilden. Dies geschieht für alle

Komponenten der Kartenvektoren M_j. Jede Komponente der gespeicherten Vektoren M_j stellt einen Prozess- oder Testparameter dar. Aus der Komponentenkarte kann man ersehen, wie sich ein einzelner Parameter über die Karte verteilt. Es wird immer nur eine bestimmte Komponente k aller gespeicherten Vektoren M_j betrachtet, deren Grösse ausgegeben und in verschiedenen Farbtönen angezeigt wird. Durch Vergleich der Komponentenkarten werden Zusammenhänge zwischen den einzelnen Komponenten erkannt. Man zieht daraus Schlussfolgerungen über die Korrelation zwischen Prozess- und Testparameter.

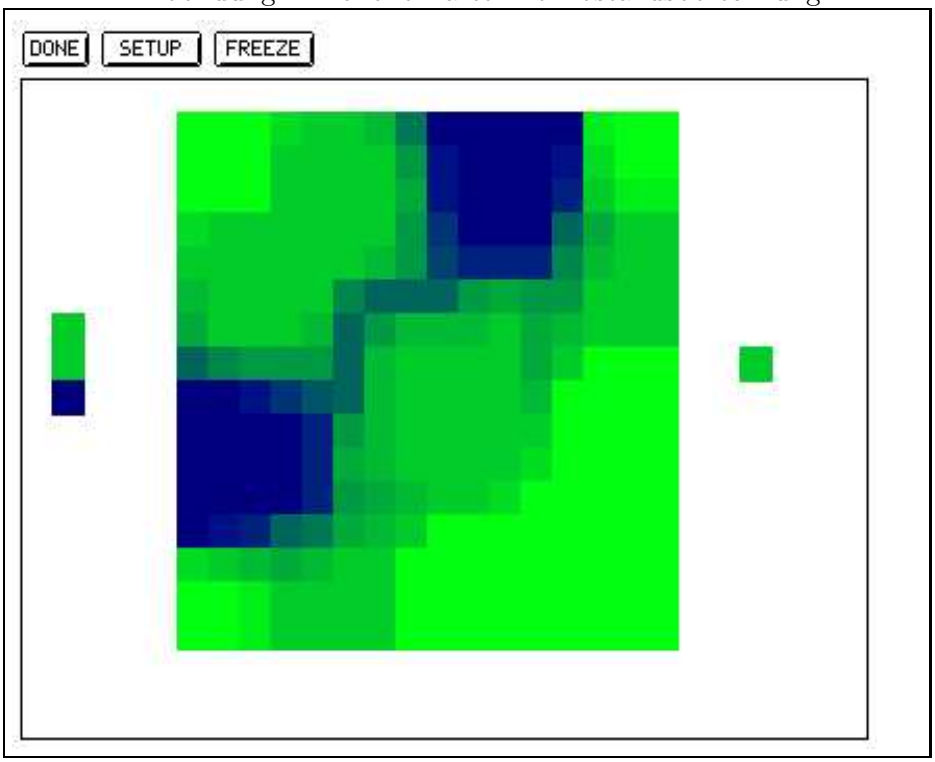
Zuallererst empfiehlt sich die Skalierung des Displays (in OPTIONS/GRID WIDTH) vom Standard 37 auf ca. 15 herunterzusetzten. Dies hat vor allem optische Gründe. Im Display selektiert man mit der Maus die Kohonen Units und tippt U S F A (Unit Set Function Activation). Um die Komponentenkarten zu aktivieren wählt man *Act_Compound*. Danach deselektiert man die Units wieder.
Im Remote Panel bei MAP selektiert man die verschiedenen Karten. Die Vorgehensweise entspricht dabei genau der Selektion verschiedener Patterns.

Abbildung 1: Remote Panel für Kohonen Funktionen



Intern passiert bei der Aktivierung der k.ten Komponentenkarte folgendes: Die Gewichte μ_{kj} werden zur aktuellen Aktivierung der Kohonen Units. Die Vektoren der Karte selbst ändern sich bei diesem Vorgang nicht, lediglich die Aktivierung!

Abbildung 2: Kohonenkarte mit Abstandsberechnung



Euklidsche Abstandsbestimmung eines Muster
Gibt man die Prozessparameter der gegenwärtig im Produktionsprozess stehenden Bauelemente auf eine bereits angelernte Karte, und bestimmt zu diesem das ähnlichste Element, braucht man nur noch festzustellen, ob sich das gefundene Element in einem Bereich der Karte befindet, wo gute Vektoren liegen. Wenn ja, kann man erwarten, dass die Produktionslinie gute Ergebnisse liefern wird. Ist dies nicht der Fall, kann man Hinweise auf die zu verbessernden Produktionsfaktoren aus der Karte entnehmen.
Im Display selektiert man die Kohonen Units und tippt U S F A. Um die euklidsche Abstandsfunktion zu aktivieren, wählt man *Act_Euclid*. Danach deselektiert man die Units wieder.
Im Remote Panel wählt man das gewünschte Pattern und STEP berechnet den euklidschen Abstand zum aktuellen Pattern und aktiviert damit die Units auf der Karte. Die Vektoren der Karte selbst ändern sich dabei nicht, so daß mit der Karte immer noch weitergearbeitet werden kann. Wer viele Muster zu bearbeiten hat, drückt den Button PSTEP (Pattern Step), jetzt braucht man nach der Wahl der Muster den STEP Button zur Berechnung nicht mehr drücken.
Achtung: Die Funktion der euklidschen Abstandsberechnung normiert die Mustervektoren zur Berechnung nicht!

5 Einschränkungen

Beim Arbeiten im SNNS mit Komponentenkarten und dem Kohonen-Lernalgorithmus gibt es verschiedene Dinge zu beachten:
Der Kohonenalgorithmus arbeitet topologisch orientiert, was zur Folge hat,

daß die Nachbarschaftsbeziehungen sowie die Anordnung der Links für ein korrektes Arbeiten wichtig sind. Der Lernalgorithmus, sowie alle Tools zur Generierung und Anzeige verarbeiten nur Netzwerke mit einer bestimmten Numerierung der Units und Reihenfolge der Links. Aus demselben Grund kann man zur Zeit noch keine mit bignet erzeugten Netze verwenden.

Literatur

[1] T. Kohonen, *Self-Organisation and associative Memory*, Springer Verlag

[2] *SNNS User Manual version 2.0*