Klassifizierung mit Self-Organizing Maps

Gernot Walzl
Technische Universität Graz, Österreich
gernot.walzl@student.tugraz.at

2008-01-23

Zusammenfassung

Dieser Artikel beschreibt, wie Self-Organizing Maps zur Klassifikation eingesetzt werden. Aufbauend auf einem einfachen Klassifikator mit einer einzelnen Self-Organizing Map wird die Hierarchische Clusterbildung eingesetzt um Schwachstellen des einfachen Klassifikators mit zusätzlichen Self-Organizing Maps zu verbessern.

Inhaltsverzeichnis

1 Einleitung

Maschinelles Lernen beschreibt, wie in künstlicher Weise Erfahrungen in Informationen umgewandelt werden können. Um dies zu erreichen, muss das System Verallgemeinerungen aus den vorgegebenen Lernbeispielen erkennen. Damit wird es möglich, Vorhersagen zu machen. Diese können einerseits ein bestimmter Output sein, um ein vorgegebenes Ziel zu erreichen oder eine Klassifizierung bestimmter Objekte, welche sich aus den Zusammenhängen der Beispiele ergibt. Wie hier schon zu erkennen ist, findet das Denken über dieses Thema bereits in der Einteilung von Kategorien und Klassen statt, deren Inhalt sich durch Ähnlichkeiten der Struktur, dem Muster widerspiegelt. Diese Klassifikation ist auch Hauptaugenmerk des maschinellen Lernens, welches sich in zwei größere Kategorien gliedern lässt:

Als **unüberwachtes Lernen** bezeichnet man Lernverfahren ohne vorher bekannte Zielwerte. Für eine Klassifizierung bedeutet dies, dass meist die Anzahl der Klassen, zu welcher eine Einteilung passen muss, im Vorhinein festgelegt wird. Die Wahl, welche Eigenschaft zu welcher Klasse gehört, bleibt dem Algorithmus überlassen.

Beim **überwachten Lernen** wird vorgeschrieben, zu welcher Art bzw. Klasse ein bestimmtes Merkmal zählt. Ein einfaches Beispiel hierfür wäre, einen Ball fangen zu lernen. Es ist gut, ihn zu fangen und schlecht, den Ball fallen zu lassen. Dieses Wissen muss vorher definiert werden. Die Erfahrungen, die damit gesammelt werden, wirken auf die Datenstruktur zurück.

Die **Self-Organizing Map** (SOM) von [Kohonen 01] ist prinzipiell eine effiziente Weise, den nichtlinearen statistischen Zusammenhang mehrerer hochdimensionaler Vektoren in eine einfache geometrische Anordnung, einem Netz von generalisierenden Knoten (Units) abzubilden. Diese geometrische Anordnung ist normalerweise zweidimensional und eignet sich hervorragend zur Visualisierung der höherdimensionalen Vektoren. Die Datenstruktur basiert prinzipiell auf einem unüberwachten Lernverfahren. Um die Self-Organizing Map zur Klassifikation verwenden zu können, ist jedoch eine überwachte Klassenzuordnung erforderlich.

In dieser Arbeit wird aufbauend anhand grundlegender Beispiele beschrieben, wie die Self-Organizing Map zur Klassifizierung verwendet werden kann.

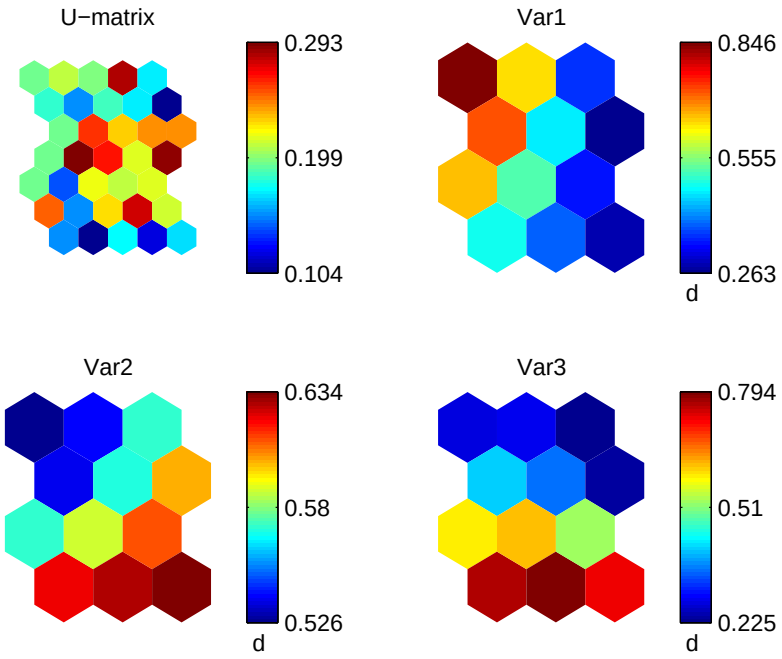
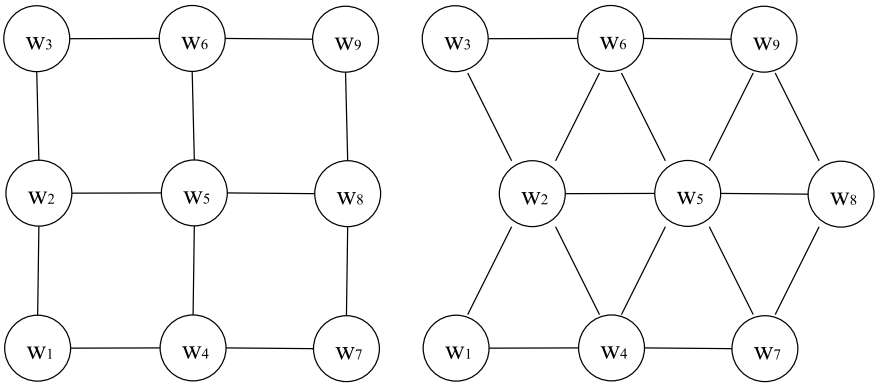


Abbildung 1: Visualisierungsbeispiel von [SOM Toolbox for Matlab 2005]

2 Self-Organizing Map

Eine Self-Organizing Map besteht aus einer Anordnung von einzelnen, untereinander verknüpften Units. Die einfachste Form der Verknüpfung ist eine rechteckige Karte. Dies ist einfach vorzustellen, anhand eines karierten Blatt Papiers, auf dem bei jedem Kreuzungspunkt der Linien eine Unit sitzt. Die Linien repräsentieren hier die Nachbarschaftsbeziehungen, wobei nur die Positionsvektoren r ausschlaggebend sind. Eine alternative Nachbarschaftsbeziehung stellt die hexagonale Anordnung dar, welche Analogien zu einer Bienenwabenanordnung aufweist.

Eine dieser Units besteht aus einer Ansammlung von Werten, einem Gewichtsvektor w. Die Dimension des Vektors ist bei allen Units gleich groß.



(a) rechteckig (b) hexagonal

Abbildung 2: Nachbarschaftsbeziehungen der Units

2.1 Lernvorgang

Entweder müssen alle Lernbeispiele (Trainingsvektoren) bekannt sein, oder die Höhe und Breite der Karte muss vorher definiert werden (siehe Kapitel 2.2). Hier wird die Höhe und Breite als Parameter vordefiniert und die Gewichtsvektoren der Units mit zufälligen Werten initialisiert.

Um einen Trainingsvektor zu lernen, wird die beste dazu passende Unit gesucht. Die am besten passende kann jene mit dem geringsten Abstand eines Distanzmaßes, wie der Manhatten-Distanz, der euklidische Distanz oder des eingeschlossenen Winkels des Gewichtsvektors der Unit w zum Trainingsvektor v sein. Die Berechnung des geringstem eingeschlossenen Winkels oder dem Skalarprodukt ist beispielsweise bei der Dokumentenklassifikation hilfreich. Für die üblichen Anwendungsgebiete wird die euklidische Distanz

d des Gewichtsvektors w zum Trainingsvektor v berechnet.

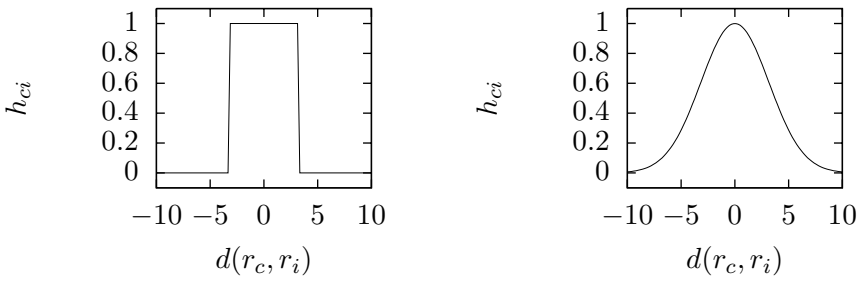
$$d(w,v) = |w - v| = \sqrt{\sum_{i=1}^{n} (w_i - v_i)^2} \tag{1}$$

Ist die Best Matching Unit (BMU) gefunden, so wird der Abstand (euklidische Distanz) mit Hilfe einer Anpassungsfunktion verringert. Anhand der Nachbarschaftsbeziehungen werden benachbarte Units ebenfalls adaptiert. Je weiter entfernt sie von der BMU sind, desto geringer soll sich die Anpassungsfunktion auf deren Gewichtsvektoren auswirken. Aus diesem Grund eignet sich die Gauß'sche Glockenfunktion ideal für diese Anforderung. Folgende Formel beschreibt diesen Zusammenhang zwischen der BMU c und einer beliebigen Unit i.

$$h_{ci}(t) = \alpha(t) * e^{-\frac{|r_c - r_i|^2}{2\sigma^2(t)}} \tag{2}$$

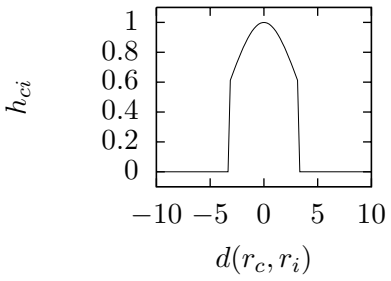
Der Parameter $\alpha(t)$ beschreibt die Geschwindigkeit des Lernens, die Lernrate. Mit der Standardabweichung $\sigma(t)$ kann man berücksichtigen, wie stark die benachbarten Units beeinflusst werden. Die Ergebnisse aller Nachbarschaftsbeziehungen können mit der diagonalsymmetrischen Kernelmatrix H beschrieben werden. Der Parameter $\sigma(t)$ wird auch als Radius des Kernels bezeichnet.

Neben der Gauß'schen Glockenfunktion kann der Kernel vereinfacht beschränkt werden oder nur aus einer "Blase" bestehen.

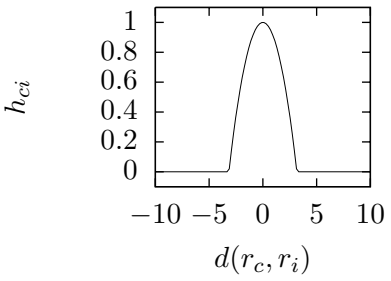


(a) bubble

(b) gaussian



(c) cutgauss



(d) ep

Abbildung 3: Verschiedene Kernel (Radius = $\sqrt{10}$)

2.1.1 Sequential Training

Der sequentielle Lernvorgang ist laut [Kohonen 01] der originale inkrementelle SOM Algorithmus. Er geht davon aus, dass sich die Gewichtsvektoren w mit einer gewissen Geschwindigkeit, der Lernrate α, den Trainingsdaten v anpassen sollen. Die Zeit t wird hierbei für jeden Berechnungsschritt ganzzahlig inkrementiert. Hier müssen im Vorhinein nicht alle Trainingsbeispiele zur Verfügung stehen, sondern können nacheinander auf die Self-Organizing Map einwirken.

$$w_i(t+1) = w_i(t) + h_{ci}(t) * (v - w_i(t)) \tag{3}$$

2.1.2 Batch Training

Hier ist es sinnvoll, den Grenzwert w_i^* der Gleichung 3 zu verstehen. Angenommen, es stehen alle Trainingsbeispiele zur Verfügung und der Grenzwert konvergiert bei $t \to \infty$, dann bedeutet dies, dass $w_i(t+1)$ und $w_i(t)$ sich, bei fortschreitender Zeit, immer ähnlicher werden. Die Änderung der Gewichtsvektoren wird somit immer geringer. Dadurch ist es möglich, die finalen Werte der Gewichtsvektoren zu errechnen.

Für ein einziges Trainingsbeispiel v gilt:

$$\begin{aligned} \lim_{t\to\infty} w_i(t+1) - w_i(t) &= \lim_{t\to\infty} h_{ci}(t) * (v - w_i(t)) \\ 0 &= v - w_i^* \\ w_i^* &= v \end{aligned}$$

Für mehrere Trainingsvektoren gilt:

$$w_i^* = \frac{\sum_j n_{ij} v_j}{\sum_j n_{ij}} \tag{4}$$

$$N := \begin{pmatrix} n_{11} & \cdots & n_{1j} \\ \vdots & \ddots & \vdots \\ n_{i1} & \cdots & n_{ij} \end{pmatrix}$$

$$N = H * P \tag{5}$$

Die Kernelmatrix H errechnet sich aus h_{ci} der Gleichung 2, wobei die Lernrate bei unendlicher Zeit irrelevant ist ($\alpha = 1$).

Die BMU-Matrix P repräsentiert eine Tabelle, die darstellt, welcher Gewichtsvektor bei welchem Trainingsbeispiel die BMU ist bzw. welcher Gewichtsvektor getroffen wurde. Die Zeilenanzahl der BMU-Matrix entspricht der Anzahl der Gewichtsvektoren. Die Spaltenanzahl entspricht der Anzahl der Trainingsbeispiele. Ein Wert dieser Matrix ist 1, wenn w_i die BMU von v_j ist und 0, wenn dies nicht zutrifft.

Durch Anpassung der Gewichtsvektoren kann es vorkommen, dass sich die BMU-Matrix ändert. In solch einem Fall muss die Berechnung wiederholt werden. Üblicherweise geschieht die Berechnung in mehreren Schritten, wobei bei jedem Iterationsschritt der Radius der Anpassungsfunktion verringert wird.

2.1.3 Vereinfachtes Berechnungsbeispiel

Um den Batch-Lernvorgang anhand eines Beispieles zu verstehen, werden zur Vereinfachung die Nachbarschaftsbeziehungen außer Kraft gesetzt. Dies entspricht einem Bubble-Kernel mit einem Radius von 0. Die Kernelmatrix ist hier eine Einheitsmatrix ($H = I$). Dadurch wird es möglich, das Ergebnis schon im Vorhinein erkennen zu können:

Wenn es keine Nachbarschaftsbeziehungen gibt, müssen die Gewichtsvektoren zum arithmetischen Mittelwert der getroffenen Trainingsbeispiele werden.

Gegeben:

$$W_{RAND} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 4 & 5 \end{pmatrix}, V = \begin{pmatrix} 2 & 3 \\ 2 & 2 \\ 1 & 0 \\ 3 & 2 \end{pmatrix}, H = I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Die BMU-Matrix ergibt sich aufgrund der geringsten Abstände zu:

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$N = H * P = I * P = P$$

$$\begin{aligned} w_i^* &= \frac{\sum_{j=1}^{4} n_{ij} v_j}{\sum_{j=1}^{4} n_{ij}} \\ w_1^* &= v_3 &&= \begin{pmatrix} 1 & 0 \end{pmatrix} \\ w_2^* &= \frac{v_1 + v_2 + v_4}{3} &&= \begin{pmatrix} 2,33 & 2,33 \end{pmatrix} \end{aligned}$$

$$W^* = \begin{pmatrix} 1 & 0 \\ 2,33 & 2,33 \\ 4 & 5 \end{pmatrix}$$

2.2 Kartengröße

Sind alle Trainingsbeispiele zu Beginn bekannt, so kann die optimale Breite und Höhe der Karte mit einer heuristischen Berechnung abgeschätzt werden.

2.2.1 Anzahl der Units

Die Anzahl der Units m hängt, nach der "guten Schätzung"
der [SOM Toolbox for Matlab 2005] nur von der Anzahl der Trainingsbeispiele l ab. Dies schlägt sich in folgender Formel nieder:

$$m = \lceil 5 * \sqrt{l} \rceil \tag{6}$$

2.2.2 Verhältnis zwischen Breite und Höhe

In das Verhältnis zwischen Breite und Höhe fließt die Gestalt der Trainingsdaten ein. Laut der [SOM Toolbox for Matlab 2005] errechnet sich dieses Verhältnis aus den beiden größten Eigenwerten der Autokovarianzmatrix der Dimensionen der Trainingsvektoren.

Die Berechnung der Autokovarianz γ zweier Vektoren u, v lautet:

$$\gamma(u, v) = E[(u - \overline{u})(v - \overline{v})] = \frac{1}{n} \sum_{i=1}^{n} (u_i - \overline{u})(v_i - \overline{v}) \tag{7}$$

Die Autokovarianzmatrix A erhält man, indem die Ergebnisse der Autokovarianz in die jeweiligen Positionen eingetragen werden. Diese Matrix ist immer diagonalsymmetrisch.

Um die Eigenwerte der Autokovarianzmatrix zu erhalten, muss das Eigenwertproblem gelöst werden:

$$det(A - \lambda I) = 0 \tag{8}$$

Das [JAMA: Java Matrix Package 2005] ist in der Lage diese Gleichung zu lösen und liefert somit alle Eigenwerte zurück. Aus den beiden größten Eigenwerten λ_1, λ_2 ergibt sich das Verhältnis zwischen Breite und Höhe der Karte.

Bei hexagonaler Anordnung gilt:

$$\text{Breite} = \left\lfloor \sqrt{\frac{m}{\sqrt{\frac{\lambda_1}{\lambda_2}}} \sqrt{\frac{3}{4}}} + 0,5 \right\rfloor \tag{9}$$

Bei rechteckiger Anordnung gilt:

$$\text{Breite} = \left\lfloor \sqrt{\frac{m}{\sqrt{\frac{\lambda_1}{\lambda_2}}}} + 0,5 \right\rfloor \tag{10}$$

$$\text{Höhe} = \left\lfloor \frac{m}{\text{Breite}} + 0,5 \right\rfloor \tag{11}$$

2.2.3 Berechnungsbeispiel

Dies ist ein grundlegendes Beispiel. Die Eigenwertzerlegung fällt unter Umständen komplexer aus.

Gegeben: hexagonale Anordnung,

$$V = \begin{pmatrix} 2 & 3 \\ 2 & 2 \\ 1 & 0 \\ 3 & 2 \end{pmatrix}$$

Anzahl der Units:

$$m = \left\lceil 5 * \sqrt{4} \right\rceil = 10$$

Verhältnis zwischen Breite und Höhe:

$$\begin{aligned} \gamma(u,v) = & \quad \tfrac{1}{4} \sum_{i=1}^{4} (u_i - \overline{u})(v_i - \overline{v}) \\ \gamma(1,1) = & \quad \tfrac{1}{4} \left((2-2)^2 + (2-2)^2 + (1-2)^2 + (3-2)^2 \right) = 0,5 \\ & \vdots \end{aligned}$$

$$A = \begin{pmatrix} 0,5 & 0,5 \\ 0,5 & 1,19 \end{pmatrix}$$

$$\begin{aligned} det(A - \lambda I) = \begin{vmatrix} 0,5 - \lambda & 0,5 \\ 0,5 & 1,19 - \lambda \end{vmatrix} & = 0 \\ (0,5 - \lambda)(1,19 - \lambda) - 0.5^2 & = 0 \\ \lambda^2 - 1,69\lambda + 0,345 & = 0 \end{aligned}$$

$$\begin{aligned} \lambda_1 &= 1,452 \\ \lambda_2 &= 0,237 \end{aligned}$$

$$\begin{aligned} \text{Breite} = & \left\lfloor \sqrt{\frac{10}{\sqrt{\frac{1,452}{0,237}}} \sqrt{0,75}} + 0,5 \right\rfloor & = 2 \\ \text{Höhe} = & \left\lfloor \tfrac{10}{2} + 0,5 \right\rfloor & = 5 \end{aligned}$$

2.3 Effizienz durch Matrizen

Viele Berechnungsschritte können durch Matrizen auf übersichtliche Weise dargestellt und implementiert werden.

2.3.1 Euklidische Distanz

Die Berechnung des Abstandes spielt für die Self-Organizing Map insoweit eine Rolle, dass zur Bestimmung der BMU die geringste Distanz gesucht werden muss. Das Minimum der quadrierten euklidischen Distanz entspricht dem geringsten Abstand. Der Vorteil hierbei liegt darin, dass die Rechenoperationen des Wurzelziehens eingespart werden können.

Die Berechnung mehrerer quadrierter Distanzen kann folgendermaßen in Matrizen umgeformt werden:

$$\begin{aligned} |w-v|^2 &= \sum_{i=1}^{n}(w_i - v_i)^2 \\ &= \sum_{i=1}^{n}(w_i^2 - 2w_iv_i + v_i^2) \\ &= \sum_{i=1}^{n} w_i^2 - 2\sum_{i=1}^{n} w_iv_i + \sum_{i=1}^{n} v_i^2 \end{aligned}$$

$$W := \begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{pmatrix} \qquad V := \begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{l1} & \cdots & v_{ln} \end{pmatrix}$$

$$W_S := \begin{pmatrix} w_{11}^2 & \cdots & w_{1n}^2 \\ \vdots & \ddots & \vdots \\ w_{m1}^2 & \cdots & w_{mn}^2 \end{pmatrix} \qquad V_S := \begin{pmatrix} v_{11}^2 & \cdots & v_{1n}^2 \\ \vdots & \ddots & \vdots \\ v_{l1}^2 & \cdots & v_{ln}^2 \end{pmatrix}$$

$$\widetilde{1} := \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}$$

$$D_S = W_S * \widetilde{1} - 2 * W * \widetilde{1} * V^T + \widetilde{1} * V_S^T \tag{12}$$

Der letzte Ausdruck dieser Formel bleibt bei einer Gewichtsanpassung konstant und muss bei einer sequentiellen Berechnung nur einmal bestimmt werden.

2.3.2 Batch Training

Hier findet sich eine Umformung der in Kapitel 2.1.2 beschriebenen Summenformeln in Matrizenmultiplikationen wieder:

$$w_i^* = \frac{\sum_j n_{ij} v_j}{\sum_j n_{ij}}$$

$$S = N * V \tag{13}$$

$$B = N * \widetilde{1} \tag{14}$$

$$w_{ij} = \frac{s_{ij}}{b_{ij}} \tag{15}$$

3 Klassifikation

Vektoren zu klassifizieren ist eines der Hauptaufgabengebiete der Self-Organizing Map. Um dieses Ziel zu erreichen, wird diese zuerst mit allen Trainingsbeispielen, unabhängig deren Klassenzugehörigkeit, trainiert. Dadurch stellen sich die Gewichte der Karte so ein, dass sie zur Klassifikation verwendet werden können. Die einzelnen Units werden anschließend mit Histogrammen analysiert, um jeder Unit eine Wahrscheinlichkeitsverteilung der Klassenzugehörigkeit zuordnen zu können.

3.1 Histogramm

Das Histogramm beschreibt, wie oft eine Unit von den Trainingsbeispielen getroffen wurde. Um es zur Klassifizierung verwenden zu können, werden die Trainingsbeispiele anhand ihrer Klassen separiert. Angenommen es gäbe 16 der Klasse 1. Es wird für jedes dieser Trainingsbeispiele die BMU bestimmt und der Zähler der Klasse 1 für diese Unit erhöht. Dadurch könnte sich folgendes Histogramm ergeben:

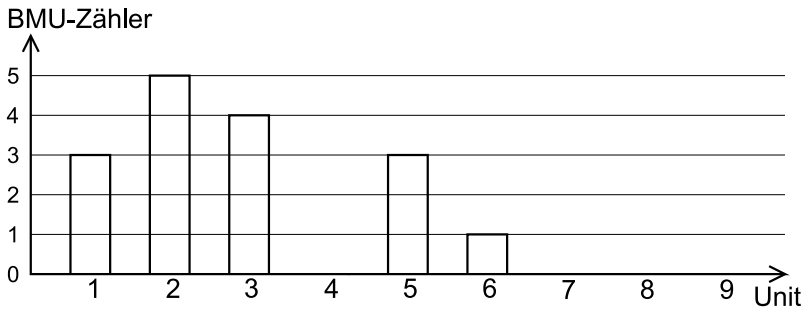


Abbildung 4: Beispiel eines Histogramms

Wie hier zu erkennen ist, hat beispielsweise Unit 8 nichts mit Klasse 1 zu tun, da sie nie von einem Trainingsbeispiel der Klasse 1 getroffen wurde.

Diese Histogramme werden für jede Klasse erstellt. Als Ergebnis zeigt sich, dass Unit 2 von 5 Trainingsbeispielen der Klasse 1, 2 Trainingsbeispielen der Klasse 2 und keinem Beispiel der Klasse 3 getroffen wurde.

Um einen Vektor zu klassifizieren, so wird zuerst seine zugehörige BMU bestimmt. Für den Fall, dass Unit 2 ausgewählt wurde, so entspricht dieser Vektor mit großer Wahrscheinlichkeit Klasse 1.

3.2 Bestimmung des relativen Fehlers

Der Klassifikator wird mit Testbeispielen, welche vorher nicht zum Lernen verwendet wurden, getestet. Jedem Testbeispiel ist klarerweise eine Klasse zugeordnet. Somit kann bestimmt werden, bei welchem Input der Klassifikator ein, von der Vorgabe abweichendes Ergebnis liefert. Aus diesen Abweichungen lässt sich eine Relative-Fehler-Matrix erstellen. Diese quadratische Matrix zeigt, zwischen welchen Klassen es Fehlinterpretationen bzw. Überlappungen gibt. Der Fehler errechnet sich aus der Differenz des Sollwertes der Anzahl der Klassen zum Istwert. Nach einer Relativierung dieser Werte erhält man die Relative-Fehler-Matrix.

3.2.1 Beispiel

Auf die genauen Werte der Trainings- bzw. Testbeispiele wird nicht eingegangen, da sie hierfür nicht erforderlich sind.

Gegeben:

$$\text{Kl. der Testbsp.} = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \\ 2 \end{pmatrix} \qquad \text{Output Klassifikator} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 2 \\ 2 \end{pmatrix}$$

Die Klassenmatrix zeigt die Klassen der Testbeispiele im Vergleich zum Ergebnis des Klassifikators. Hätte der Klassifikator keine Fehler gemacht, so wäre nur die Hauptdiagonale positiv befüllt.

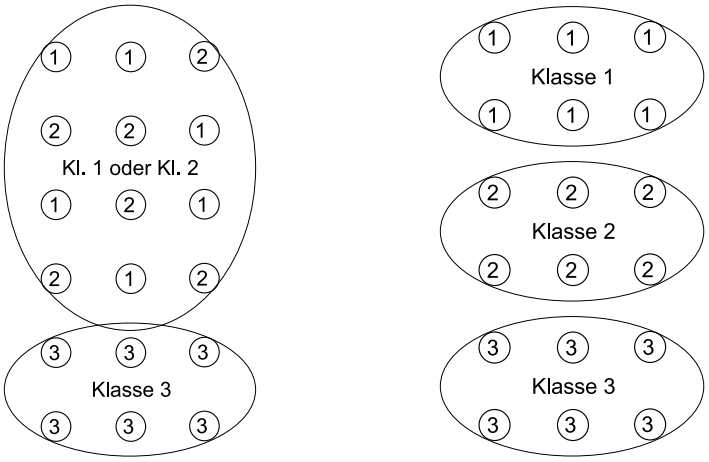
$$\begin{pmatrix} 0 & 2 \\ 1 & 2 \end{pmatrix}$$

Daraus ergibt sich die Relative-Fehler-Matrix zu:

$$\begin{pmatrix} 0 & 1 \\ 0.33 & 0 \end{pmatrix}$$

4 Hierarchisches Clustern

Das Problem des normalen Klassifikators liegt darin, dass es vorkommen kann, dass sich die Bereiche der Klassen oft stark überlappen. Sie sind in diesem Fall mit einer einfachen Self-Organizing Map nicht gut trennbar. Die Fehleranzahl ist bei überlappenden Bereichen eine erhöhte.



(a) Überlappungen (schlecht)

(b) keine Überlappungen (gut)

Abbildung 5: Schwachstellen des einfachen Klassifikators

Wie in Abbildung 5a zu sehen ist, kann diese Karte den Unterschied zwischen Klasse 1 und Klasse 2 nicht gut erkennen. Um die Schwachstelle zu verbessern, werden diese beiden Klassen in eine weitere Self-Organizing Map ausgelagert. In der übergeordneten SOM wird nun eine Referenz auf die nächste gespeichert.

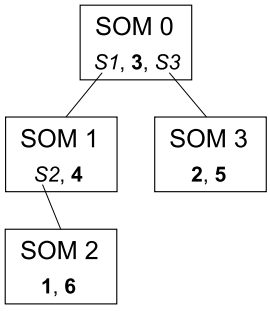


Abbildung 6: Beispiel eines SOM-Klassifizierungsbaumes

Um einen Klassifizierungsbaum zu erstellen, wird im ersten Schritt die Fehlermatrix mit einem sehr niedrigen Schwellwert erstellt. Anschließend

werden daraus Partitionierungslisten erstellt. Wenn aufgrund des geringen Schwellwertes alle Klassen als auszulagernde Partition erkannt werden, so wird der Schwellwert multiplikativ erhöht und der ganze Vorgang wiederholt.

4.1 Fehlermatrix

Mit Hilfe der relativen Fehler des einfachen Klassifikators und einem Schwellwert kann die Fehlermatrix erstellt werden. Liegt der relative Fehler über dem Schwellwert wird eine 1, ansonsten eine 0 eingetragen. Anschließend wird diese Matrix diagonalsymmetrisch gemacht, indem alle 0 auf 1 ergänzt werden.

4.1.1 Beispiel

Gegeben:

$$\text{Schwellwert} = 0.5 \qquad \text{Relative-Fehler-Matrix} = \begin{pmatrix} 0 & 1 \\ 0.33 & 0 \end{pmatrix}$$

Nach der Schwellwertoperation ergibt sich folgende Matrix:

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Diese Matrix wird anschließend diagonalsymmetrisch gemacht:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

4.2 Partitionierungslisten

Eine Partitionierungsliste PL zeigt, welche Klassen in einen zusätzlichen SOM-Klassifikator ausgelagert werden sollen.

Zu Beginn wird die erste Zeile der Fehlermatrix als PL genommen und als fertig markiert. Ist ein Spaltenwert 1, so wird die jeweilige Zeile mit der aktuellen PL ODER-verknüpft, als fertig markiert und das Ergebnis in die aktuelle PL eingetragen. Dieser Vorgang wird rekursiv wiederholt. Als Ergebnis erhält man die erste PL. Um die nächste PL zu generieren, wird überprüft, welche Zeilen der Fehlermatrix noch nicht als fertig markiert sind. Ist eine solche Zeile gefunden, so wird mit dieser begonnen, die nächste PL zu erstellen. Dieser Ablauf wiederholt sich, bis alle Zeilen als fertig markiert sind.

4.2.1 Beispiel

Gegeben:

$$\text{Fehlermatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

PL_1 wird initialisiert:

$$PL_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \qquad \text{fertig} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Die jeweilige Zeile wird ODER-verknüpft zur PL hinzugefügt:

$$PL_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix} \qquad \text{fertig} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Damit ist die erste PL bestimmt. Die nächste nicht fertige Zeile ist die zweite. Damit wird die nächste PL initialisiert und der Ablauf wiederholt.

Es ergeben sich folgende Partitionierungslisten:

$$PL_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}$$

$$PL_2 = \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}$$

Dies bedeutet, dass Klasse 1 mit Klasse 3 in einen zusätzlichen Klassifikator ausgelagert werden und Klasse 2 mit Klasse 4 in einen weiteren.

5 Anwendungsgebiete

5.1 Fangen eines Balles

In [Self Organizing Map - ein Demonstrationsbeispiel 2001] wird ein Einsatzgebiet des sequentiellen Trainings gezeigt. Die Gewichtsvektoren bestehen aus der Abfeuerungsgeschwindigkeit des Balles v_{Ball}, der Position x_{Fzg} und Geschwindigkeit des Fahrzeuges v_{Fzg} und sind somit dreidimensional.

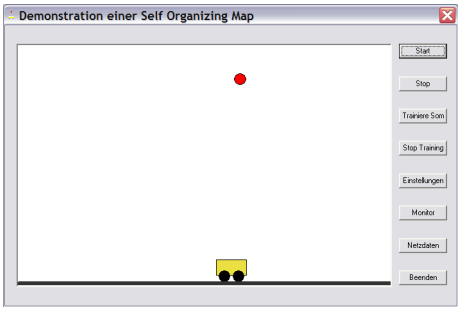
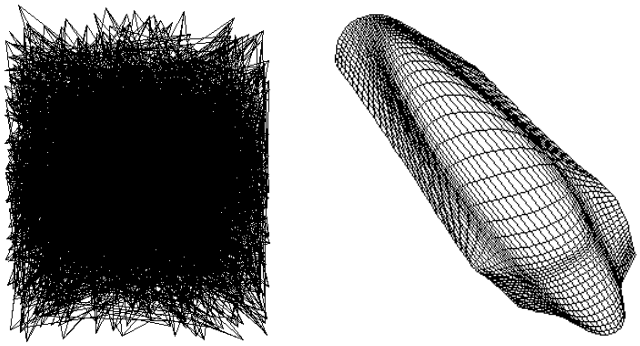


Abbildung 7: Programm zum Ballfangen

Damit das Fahrzeug aus dem Vektor der Fahrzeugposition und Ballgeschwindigkeit weiß, was es tun soll, wird zu diesen Werten die BMU bestimmt. Die resultierende Fahrzeuggeschwindigkeit bekommt man aus dem dazugehörigen Gewichtsvektor. Nach der Initialisierung ist das Verhalten zufällig. Wenn der Ball gefangen wird, ergibt sich aus den drei Werten (v_{Ball}, v_{Fzg}, x_{Fzg}) ein Trainingsvektor. Mit diesem wird die Self-Organizing Map mittels sequentiellen Trainings angepasst, wodurch das Verhalten des Fahrzeuges immer zielgerichteter wird.



(a) zufällig initialisiert (b) nach dem Lernvorgang

Abbildung 8: Gewichtsvektoren als Positionskoordinaten

5.2 Klassifikation von Netzwerkverkehr

Wie in [Payer 05] beschrieben, eignet sich die SOM gut zur Paketanalyse von Netzwerkverkehr. Die Trainingsvektoren werden aus mehreren Features zusammengesetzt. Ein Feature ist beispielsweise ein Histogramm über die Byte-Verteilung. Es ist klar, dass ein *http* Paket mit dem Standard ASCII-Zeichensatz auskommt, damit nur aus gewissen Bytes besteht und *https* Paket aufgrund der Verschlüsselung ein relativ gleichverteiltes Byte-Histogramm aufweist.

Weitere Entwicklungen auf diesem Gebiet finden sich in [Payer 08] wieder.

5.2.1 Einfacher Klassifikator

	ftp	**http**	**https**	**mysql**	**pop3**	**smtp**	**ssh**	**vnc**
ftp	**100%**	0%	0%	6,7%	20%	7,1%	0%	7,1%
http	0%	**100%**	0%	0%	0%	0%	0%	14,3%
https	0%	0%	**100%**	0%	0%	0%	0%	0%
mysql	0%	0%	0%	**73,3%**	0%	0%	0%	0%
pop3	0%	0%	0%	0%	**0%**	0%	0%	0%
smtp	0%	0%	0%	0%	80%	**92,9%**	0%	0%
ssh	0%	0%	0%	0%	0%	0%	**100%**	0%
vnc	0%	0%	0%	20%	0%	0%	0%	**64,3%**

Tabelle 1: Klassifikationswahrscheinlichkeiten des Byte-Histogramms mit Standard SOM-Klassifikator

5.2.2 Klassifikator mit hierarchischer Clusterbildung

	ftp	**http**	**https**	**mysql**	**pop3**	**smtp**	**ssh**	**vnc**
ftp	**100%**	0%	0%	6,7%	0%	0%	0%	0%
http	0%	**100%**	0%	0%	0%	0%	0%	0%
https	0%	0%	**90,9%**	0%	0%	0%	0%	14,3%
mysql	0%	0%	0%	**73,3%**	0%	0%	0%	0%
pop3	0%	0%	0%	0%	**26,7%**	14,3%	0%	7,1%
smtp	0%	0%	0%	0%	73,3%	**85,7%**	0%	0%
ssh	0%	0%	9,1%	0%	0%	0%	**100%**	0%
vnc	0%	0%	0%	20%	0%	0%	0%	**64,3%**

Tabelle 2: Klassifikationswahrscheinlichkeiten des Byte-Histogramms mit SOMTree-Klassifikator

6 Zusammenfassung und Ausblick

Die Self-Organizing Map von [Kohonen 01] ist, im Vergleich zu konventionellen künstlichen neuronalen Netzen, einfach und schnell zu berechnen. Dies macht es zu einem äußerst effizienten Werkzeug des Maschinellen Lernens und bringt die Computer einen Schritt weiter zur Fähigkeit der Erkenntnis logischer und struktureller Zusammenhänge.

Das Verbesserungspotential der SOM besteht darin, dass die Anzahl der Gewichte im Vorhinein festgelegt werden muss. Um diese Anzahl dynamisch, während des Lernvorganges ändern zu können, ist eine Aufhebung der geometrischen Zusammenhänge sehr hilfreich. Solch einen Ansatz findet man im **Neural Gas** (NG) von [Martinetz 91]. In dieser multidimensionalen "Wolke" werden die Nachbarschaftsbeziehungen nur durch die Gewichte der Units und nicht durch eine Vorgabe definiert. Bei einem solch losen Gebilde ist es einfach, weitere Units zur Verbesserung der Fehlerrate einzufügen. Das Ganze muss jedoch mit Maß und Ziel erfolgen, da ein Überfüllen negative Auswirkungen auf das Gesamtergebnis hat. Wie man daraus die goldene Mitte trifft, wird im **Growing Neural Gas** (GNG) von [Fritzke 94] beschrieben.

Literatur

[Kohonen 01] Kohonen, Teuvo: "Self-Organizing Maps"; Springer, Berlin (2001), ISBN-10: 3-540-67921-9

[Payer 05] Payer, Udo; Teufl, Peter; Lamberger, Mario: "Traffic Classification using Self-Organizing Maps"; INC 2005 Fifth International Networking Conference Workshops, 05-07 July 2005, Samos Island, Greece

[SOM research 2005] `http://www.cis.hut.fi/research/som-research/` (2005-03-17, Department of Computer Science and Engineering, Helsinki University of Technology)

[SOM Toolbox for Matlab 2005] `http://www.cis.hut.fi/projects/somtoolbox/` (2005-03-23, Department of Computer Science and Engineering, Helsinki University of Technology)

[Self Organizing Map - ein Demonstrationsbeispiel 2001] `http://www.informatik.htw-dresden.de/~iwe/Belege/Schneider/som.html` (2001-03, Hochschule für Technik und Wirtschaft Dresden)

[JAMA: Java Matrix Package 2005] `http://math.nist.gov/javanumerics/jama/` (2005-07-13, National Institute of Standards and Technology)

[Wikipedia: Maschinelles Lernen 2007] `http://de.wikipedia.org/wiki/Maschinelles_Lernen` (2007-10-10, Wikipedia)

[Wikipedia: Selbstorganisierende Karte 2007] `http://de.wikipedia.org/wiki/Selbstorganisierende_Karte` (2007-10-10, Wikipedia)

[Martinetz 91] Martinetz, T. M.; Schulten, K. J.: "neural-gas network learns topologies" Artificial Neural Networks, pages 397-402. North-Holland, Amsterdam, 1991

[Fritzke 94] Fritzke, Bernd: "Fast learning with incremental RPF networks" Neural Processing Letters, 1(1):-5, 1994b

Angenommen, aber noch nicht publiziert:

[Payer 08] Payer, Udo; Teufl, Peter; et al.: "InFeCT - Network Traffic Classification"; ICN 2008 "Networking Conference", 13-18 April 2008, Cancun, Mexico