Aufgabe: Leiten Sie ein neuronales Verfahren zur Gesichtserkennung her, indem Sie die Gesichtsfläche in eine endliche Anzahl elementarer Grunddreiecke zerlegen, deren Eckpunkte zueinander in einem für jeden Menschen eigentümlichen Abstandsverhältnis stehen. Berechnen Sie sodann anhand eines Backpropagation-Algorithmus die Gewichte für ein zweistufiges neuronales Netz ohne verdeckte Schicht.

Lösung: Der Idee einer Gesichtserkennung liegt der Gedanke zugrunde, daß es mit Ausnahme von eineiigen Zwillingen keine zwei gleichen Gesichter gibt. Vergleichen wir zwei Gesichter miteinander, so stellen wir fest, daß Menschen unterschiedlich lange Nasen, unterschiedliche Augenabstände oder unterschiedliche Mundbreiten haben. Diese Merkmale werden biometrische Daten genannt. Ein Beispiel dafür liefert Abb. 1.

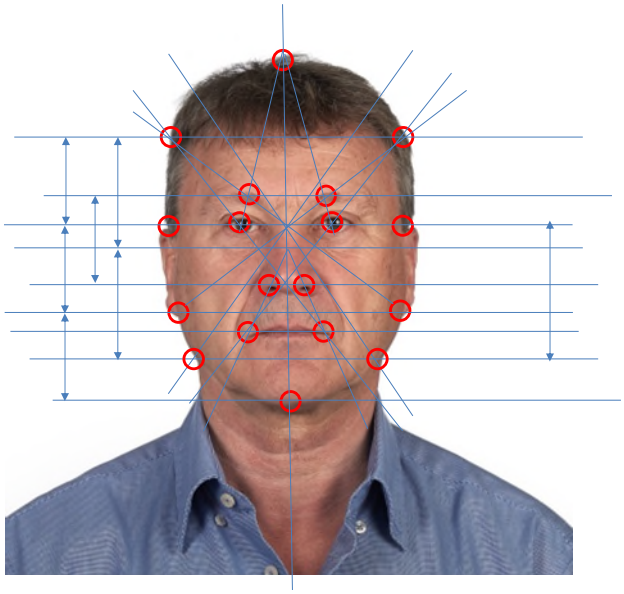

Abbildung 1. Unveränderliche biometrische Linien im Gesicht

So lassen sich etwa anhand markanter Punkte wie Augen- und Nasenmitte, Kinnspitze etc. oder charakteristischer Abmessungen wir Ohrenabstand, Nasen- oder Oberlippenlänge Linien durchs Gesicht ziehen, die ein biometrisches Profil eines Menschen ergeben und somit ein unveränderliches Körpermerkmal darstellen. Die Idee ist nun: Wenn man eine charakteristische Abmessung, etwa die Distanz der Augenlinie zur Nasenlinie als Normierungsgröße verwendet, werden sich sämtliche Menschen in mindestens einem, aber höchstwahrscheinlich in mehreren, möglicherweise sogar vielen Merkmalen von der Referenzperson unterscheiden.

Wählen wir nun eine endliche Anzahl charakteristischer Punkte im Gesicht (rote Kreise in Abb. 1), so lassen sich aus jeweils 3 Punkten Dreiecksflächen bilden, die eine eindeutig definierte Flächennormale besitzen, siehe Abb. 2. Die Darstellung eines Gesichts ist nichts anderes als die Projektion einer räumlich gekrümmten Oberfläche in die Bildebene, wobei zwischen der dreidimensionalen Flächennormalen $\mathbf{A}$ und der Flächennormalen der projizierten Fläche $\mathbf{A}_n$ der Winkel θ liegt. Jedes Flächenelement auf dem Foto ist also um den Kosinus des Winkels,

den der Normalenvektor auf der dreidimensionalen Gesichtsoberfläche mit dem Lot senkrecht zur Bildebene einnimmt, kleiner (Abb. 2).

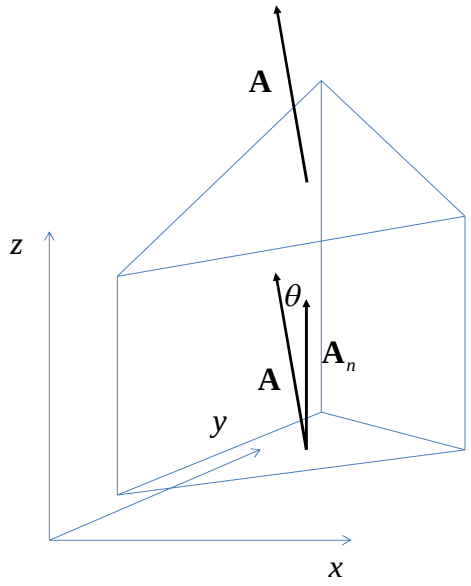

Abbildung 2. Die Flächennormale auf einer gekrümmten Oberfläche und ihre Projektion in die Bildebene

Da unser Gesicht gekrümmt ist, müssen wir es zunächst in eine ausreichende Zahl von Flächenelementen zerlegen. Als Zerlegung der Gesichtsfläche wählen wir ohne Beschränkung der Allgemeinheit[1] die in Abb. 3 dargestellte Anordnung aus 25 Dreiecken, die nahtlos aneinander anschließen.

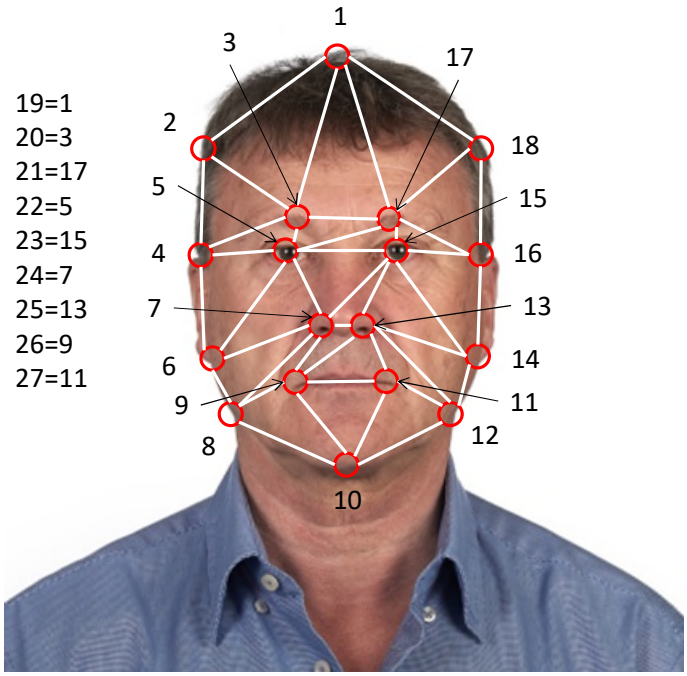

Abbildung 3. Zerlegung einer Gesichtsfläche in Elementardreiecke

[1] Die vorgestellte Zerlegung ist insoweit nur beispielhaft, und es ist nicht gesagt, daß es nicht noch bessere Möglichkeiten der Zerlegung gibt.

Dabei ist jeder der insgesamt 27 Eckpunkte wenigstens einmal Ausgangs- oder Endpunkt einer der 25 Dreiecksflächen $A_{i,i+1,i+2}$, die in aufsteigender Folge wie folgt angeordnet werden können:

$$A_{123}, A_{234}, A_{345}, \ldots, A_{i,i+1,i+2}, \ldots, A_{m,m+1,m+2}.$$

Um also eine Fläche in m Dreiecke zerlegen zu können, benötigen wir $m+2$ Eckpunkte. In unserem Fall ist $m = 25$. Hinsichtlich der Zerlegung ist auch noch wichtig, diese so zu wählen, daß eine fortlaufende Reihe von Punkten entsteht, bei denen möglichst keiner ausgelassen wird oder einer nicht Eckpunkt eines Grunddreiecks ist. Doppelzählungen sind ausdrücklich erlaubt, bisweilen sogar unvermeidbar. Abb. 3 zeigt eine Zerlegung mit 18 Eckpunkten und 9 weiteren Punkten, die doppelt vergeben wurden. Da unsere Ausführungen nur exemplarisch sind, beschränken wir unsere Untersuchungen auf zwei Testpersonen (Abb. 4), deren Gesichtsflächen nach demselben Muster zerlegt wurden. Die linke dieser Personen entspricht der Referenzperson, bis auf den Winkel der Kopfdrehung, den es nachfolgend zu ermitteln gilt.

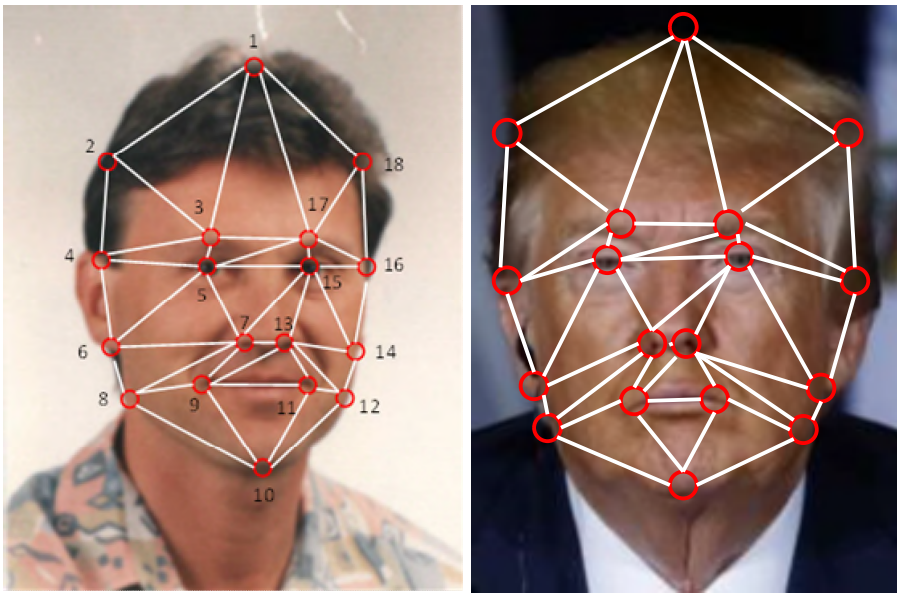

Abbildung 4. Die zu vergleichenden Ist-Bilder

Sämtliche Teildreiecke der Bildebene addieren sich nun nach folgender Formel zu einer Gesamtfläche

$$A = \sum_{j=1}^{m} A_{j,j+1,j+2},$$

die wir als Normierungsgröße brauchen, weil wir die Neuronen zu Vergleichszwecken als Werte kleiner oder gleich 1 behandeln müssen. Für die normierten Dreiecksflächen gilt dann der Ausdruck

$$a_{j,j+1,j+2} \equiv \frac{A_{j,j+1,j+2}}{A}.$$

Zur Berechnung einer Dreiecksfläche im dreidimensionalen Raum benötigen wir die Flächennormale, die sich als Kreuzprodukt der beiden Abstandsvektoren $\Delta\mathbf{r}_{ij}$ und $\Delta\mathbf{r}_{ik}$ von einem gemeinsamen Eckpunkt *i* zu jedem der zwei anderen Eckpunkte *j* und *k* im Dreieck darstellen läßt, d.h.

$$\mathbf{A}_{ijk} = \frac{1}{2}\Delta\mathbf{r}_{ij} \times \Delta\mathbf{r}_{ik} = \frac{1}{2}\begin{pmatrix}\Delta x_{ij}\\ \Delta y_{ij}\\ \Delta z_{ij}\end{pmatrix} \times \begin{pmatrix}\Delta x_{ik}\\ \Delta y_{ik}\\ \Delta z_{ik}\end{pmatrix} = \frac{1}{2}\begin{pmatrix}\Delta y_{ij}\Delta z_{ik} - \Delta z_{ij}\Delta y_{ik}\\ \Delta z_{ij}\Delta x_{ik} - \Delta x_{ij}\Delta z_{ik}\\ \Delta x_{ij}\Delta y_{ik} - \Delta y_{ij}\Delta x_{ik}\end{pmatrix}.$$

Für ebene Vektoren in der *x*-*y*-Ebene verbleibt von diesem Vektor nur die *z*-Komponente

$$A_{ijk} = \frac{1}{2}\left|\Delta\mathbf{r}_{ij} \times \Delta\mathbf{r}_{ik}\right| = \frac{1}{2}\left|\begin{pmatrix}\Delta x_{ij}\\ \Delta y_{ij}\\ 0\end{pmatrix} \times \begin{pmatrix}\Delta x_{ik}\\ \Delta y_{ik}\\ 0\end{pmatrix}\right| = \frac{1}{2}\left|\Delta x_{ij}\Delta y_{ik} - \Delta y_{ij}\Delta x_{ik}\right|,$$

wobei wir Betragsstriche gesetzt haben, da Flächen auch negativ sein können. Im einzelnen gelten für die jeweiligen Abstandsdifferenzen die Relationen

$$\begin{aligned}\Delta x_{ij} &= x_j - x_i, \quad & \Delta x_{ik} &= x_k - x_i,\\ \Delta y_{ij} &= y_j - y_i, \quad & \Delta y_{ik} &= y_k - y_i,\\ \Delta z_{ij} &= z_j - z_i, \quad & \Delta z_{ik} &= z_k - z_i.\end{aligned}$$

Um einer Vielzahl möglicher Gesichter Rechnung zu tragen, führen wir außerdem für jedes einzelne ein Trainingsmuster mit dem Index *p* ein.

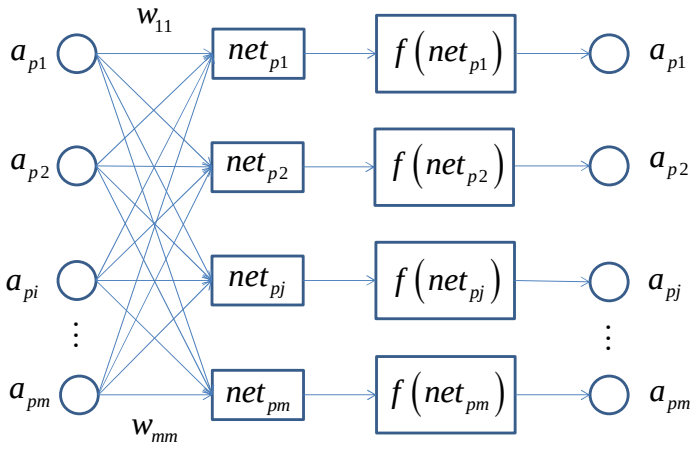

Abbildung 5. Zweistufiges Netz ohne verdeckte Schicht

Um Laufindizes einzusparen, definieren wir die Ausgangsneuronen a_{pj} sowie die Sollausgangswerte $\hat{a}_{pj}$ zunächst durch

$$a_{pj} \equiv a_{p,j,j+1,j+2} = \frac{A_{p,j,j+1,j+2}}{A_p} = \frac{\left|\Delta x_{p,j,j+1}\Delta y_{p,j,j+2} - \Delta y_{p,j,j+1}\Delta x_{p,j,j+2}\right|}{\sum_{j=1}^{m} A_{p,j,j+1,j+2}},$$

$$\hat{a}_{pj} \equiv \hat{a}_{p,j,j+1,j+2} = \frac{\hat{A}_{p,j,j+1,j+2}}{\hat{A}_p} = \frac{\left|\Delta \hat{x}_{p,j,j+1}\Delta \hat{y}_{p,j,j+2} - \Delta \hat{y}_{p,j,j+1}\Delta \hat{x}_{p,j,j+2}\right|}{\sum_{j=1}^{m} \hat{A}_{p,j,j+1,j+2}}$$

für $j = 1, 2, \ldots, m.$ In unserem zweilagigen Netz (siehe Abb. 5) ist also

a_{pi}	der Input des Neurons i in der Eingabeschicht, wobei $1 \le i \le m$,
a_{pj}	der Output des Neurons j in der Ausgabeschicht, wobei $1 \le j \le m$,
net_{pj}	der Netto-Input des Neurons j in der Ausgabeschicht aus Schicht 1, wobei $1 \le j \le m$,
w_{ij}	das Verbindungsgewicht von Neuron i nach j an der Schnittstelle.

Nachdem wir das Netz mit dem Gesicht in Abb. 3 angelernt haben, wollen wir es auf die konkreten Fälle von Abb. 4 anwenden.

$$a_{pj} = \frac{A_{pj}^{\perp}}{A_p^{\perp}} = \begin{cases} \dfrac{A_{pj}\cos\left(\theta_{pj} - \theta_p\right)}{\sum_{j=1}^{m} A_{pj}^{\perp}} & \text{für} \quad j \in \{1, 2, 3, 4, 5, 6, 7, 8\}, \\ \dfrac{A_{pj}\cos\theta_{pj}}{\sum_{j=1}^{m} A_{pj}^{\perp}} & \text{für} \quad j \in \{9, 19, 20, 21, 22, 23, 24, 25\}, \\ \dfrac{A_{pj}\cos\left(\theta_{pj} + \theta_p\right)}{\sum_{j=1}^{m} A_{pj}^{\perp}} & \text{für} \quad j \in \{10, 11, 12, 13, 14, 15, 16, 17\}, \end{cases}$$

wobei $A_{p,18}^{\perp} = 0.$ Zur expliziten Unterscheidung von den dreidimensionalen Flächenelementen haben wir die zweidimensionalen Projektionen mit $A_{pj}^{\perp}$ anstelle von A_{pj} bezeichnet. Ferner gilt

$$\hat{a}_{pj} = \frac{\hat{A}_{pj}^{\perp}}{\hat{A}_p^{\perp}} = \frac{\hat{A}_{pj}\cos\theta_{pj}}{\sum_{j=1}^{m} \hat{A}_{pj}^{\perp}}$$

mit $\hat{A}_{p,18}^{\perp} = 0$ und $\hat{A}_{pj}^{\perp}$ anstelle von $\hat{A}_{pj}$. Abb. 6 zeigt am Beispiel der Flächenelemente 1 und 17 anschaulich, wie sich die Auswirkungen einer Kopfdrehung kompensieren lassen. Das linke Halbbild weist für den Winkel $\theta = 0$ eine vollkommene Symmetrie auf, wogegen sich die Kopfdrehung im rechten Halbbild in unterschiedlich großen Flächendreiecken rechts und links der vertikalen Symmetrieachse des Gesichts auswirkt. Bei einer Linksdrehung des Kopfes müs-

sen der Projektionswinkel θ_j der Flächennormalen und der Drehwinkel θ voneinander subtrahiert werden, da die Dreiecksflächen dem Betrachter zugewandt sind und rechts von der Symmetrieachse liegen. Sind sie hingegen links von der Symmetrieachse, müssen sie addiert werden. Die Kopfdrehung der Soll-Bilder kann daher durch Mittelwertbildung von zusammengehörigen Flächenelementen rechts und links der Gesichtsmitte rechnerisch eliminiert werden. Flächen wiederum, die kein Pendant haben, bleiben ungemittelt. In Abb. 6 ist die Geometrie der Kopfbewegung aus der Draufsicht von oben dargestellt, also mit dem Gesicht nach unten. Das Verfahren wird nachfolgend beschrieben.

$$A_1 \cos(\theta_1 - \theta) + A_{17} \cos(\theta_{17} + \theta) = 2A_1 \cos\theta_1 \cos\theta$$

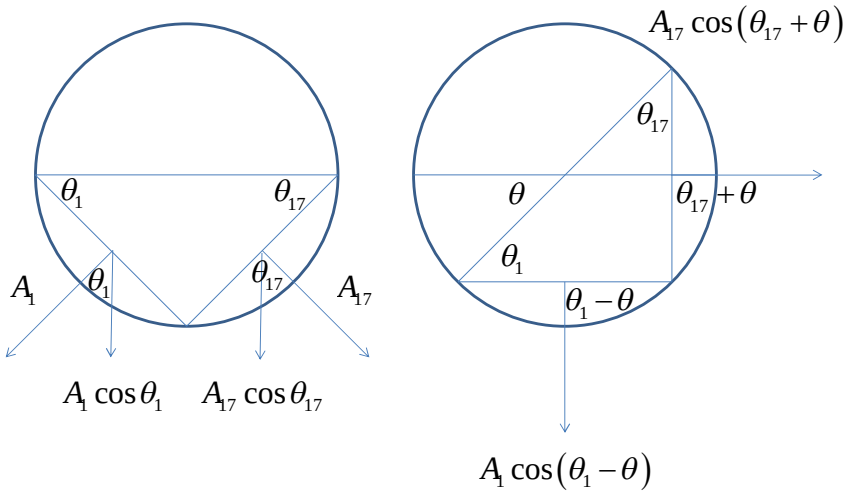

Abbildung 6. Auswirkungen und Kompensation der Kopfdrehung auf die Gesichtsfläche

Dazu lassen wir den Index p am besten weg, weil wir nur ein einziges Bild betrachten. Für $j \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ gilt im Falle der biometrischen Draufsicht $(\theta = 0)$ für die Referenzflächen

$$\hat{a}_j = \hat{a}_{18-j} = \frac{\hat{A}_j^{\perp} + \hat{A}_{18-j}^{\perp}}{2\hat{A}^{\perp}} = \frac{\hat{A}_j \cos\theta_j + \hat{A}_{18-j} \cos\theta_{18-j}}{2\hat{A}^{\perp}} = \frac{\hat{A}_j \cos\theta_j}{\hat{A}^{\perp}}$$

bzw.

$$\hat{a}_j = \frac{\hat{A}_j^{\perp}}{\hat{A}^{\perp}} = \frac{\hat{A}_j \cos\theta_j}{\hat{A}^{\perp}} \quad \text{für} \quad j \in \{9, 18, 19, 20, 21, 22, 23, 24, 25\}.$$

Weil wir davon ausgehen können, daß, wie in Abb. 6. dargestellt, $\theta_j = \theta_{18-j}$ und $A_j = A_{18-j}$, können die symmetrischen Flächen für $j \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ im allgemeinen Fall einer leichten Schrägsicht gemittelt werden, d.h.

$$a_j = a_{18-j} = \frac{A_j^{\perp} + A_{18-j}^{\perp}}{2A^{\perp}} = \frac{A_j \cos(\theta_j - \theta) + A_{18-j} \cos(\theta_{18-j} + \theta)}{2A^{\perp}} = \frac{A_j \cos\theta_j \cos\theta}{A^{\perp}}.$$

Für die asymmetrischen Flächen gilt stattdessen

$$a_j = \frac{A_j^{\perp}}{A^{\perp}} = \frac{A_j \cos\theta_j \cos\theta}{A^{\perp}} \quad \text{für} \quad j \in \{9, 18, 19, 20, 21, 22, 23, 24, 25\}.$$

Wenn wir beide Gleichungen durcheinander dividieren, erhalten wir Relationen, aus denen wir den Drehwinkel θ bestimmen können,

$$\cos\theta = \frac{A_j^{\perp} + A_{18-j}^{\perp}}{\hat{A}_j^{\perp} + \hat{A}_{18-j}^{\perp}} \frac{\hat{A}_j}{A_j} \quad \text{für} \quad j \in \{1, 2, 3, 4, 5, 6, 7, 8\},$$

$$\cos\theta = \frac{A_j^{\perp}}{\hat{A}_j^{\perp}} \frac{\hat{A}_j}{A_j} \quad \text{für} \quad j \in \{9, 18, 19, 20, 21, 22, 23, 24, 25\}.$$

Für normierte Gesichtsflächen ist $\hat{A}_j = A_j$ usw. Damit können die Winkel direkt mittels der Relationen

$$\cos\theta = \frac{a_j A^{\perp}}{\hat{a}_j \hat{A}^{\perp}} = \frac{A_j^{\perp} + A_{18-j}^{\perp}}{\hat{A}_j^{\perp} + \hat{A}_{18-j}^{\perp}} \quad \text{bzw.} \quad \cos\theta = \frac{A_j^{\perp}}{\hat{A}_j^{\perp}}$$

berechnet werden. Zur Normierung der Bilder verwenden wir den Abstand

$$h_0 = \left| \frac{\hat{\mathbf{r}}_{15} + \hat{\mathbf{r}}_5}{2} - \frac{\hat{\mathbf{r}}_{13} + \hat{\mathbf{r}}_7}{2} \right| = \frac{1}{2} \sqrt{\left(\hat{x}_{15} + \hat{x}_5 - \hat{x}_{13} - \hat{x}_7\right)^2 + \left(\hat{y}_{15} + \hat{y}_5 - \hat{y}_{13} - \hat{y}_7\right)^2}$$

zwischen Augen- und Nasenmitte aus dem Referenzbild. Im tatsächlich vorgelegten Sollbild lautet dieser Abstand

$$h = \left| \frac{\mathbf{r}_{15} + \mathbf{r}_5}{2} - \frac{\mathbf{r}_{13} + \mathbf{r}_7}{2} \right| = \frac{1}{2} \sqrt{\left(x_{15} + x_5 - x_{13} - x_7\right)^2 + \left(y_{15} + y_5 - y_{13} - y_7\right)^2}.$$

Um nun ein tatsächlich auszuwertendes Bild zu normieren, müssen alle Positionsdaten mit dem Faktor $\kappa = h_0 / h$ multipliziert werden, d.h. $\kappa < 1,$ wenn $h \geq h_0,$ und $\geq 1,$ wenn $h < h_0.$

So ergibt sich beispielsweise für die Vergleichsperson ein κ von 0,7173 und damit ein korrigierter Fehler von $E = 0{,}0012898,$ während das κ von Donald Trump gleich 1,0139 ist und somit einem Fehler von $E = 0{,}0018017$ entspricht. Donald Trump weicht also von der Vergleichsperson um 39,7 % ab, was eine deutliche Unterscheidung ermöglicht. Das Verfahren ist also für eine Gesichtserkennung[2] geeignet. Nachfolgend leiten wir noch den Trainingsalgorithmus ab.

Ausgangspunkt für die Herleitung des Backpropagation-Algorithmus ist die Minimierungsvorschrift des Lernfehlers

[2] Eigentlich Identifikation

$$E_p = \frac{1}{2}\sum_{j=1}^{m}\left(\hat{a}_{pj} - a_{pj}\right)^2$$

in einem Muster *p*. Der nun folgende Ansatz basiert auf der Überlegung, daß die Berechnung des Gradienten für die Gesamtfehlerfunktion *E* durch Summation der Gradienten über alle Fehlerfunktionen E_p der einzelnen Trainingsmuster erfolgen kann, da der Gesamtfehler *E* nichts anderes ist als die zu minimierende Summe

$$E = \sum_{p=1}^{q} E_p = \frac{1}{2}\sum_{p=1}^{q}\sum_{j=1}^{m}\left(\hat{a}_{pj} - a_{pj}\right)^2 = \min$$

aller *q* Lernfehler. Da der Lernfehler *E* abnehmend ist, d.h. $\partial E/\partial w_i < 0,$ muß für $i = 1, 2, \ldots, n$ gelten:

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i} = -\gamma \sum_{p=1}^{q} \frac{\partial E_p}{\partial w_i},$$

wobei der Proportionalitätsfaktor γ die Lernrate angibt. Die Modifikationsregel für alle *n* Verbindungsgewichte kann demnach geschrieben werden als

$$\Delta \mathbf{w} = -\gamma \nabla E(\mathbf{w}) = -\gamma\left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n}\right) = -\gamma\left(\sum_{p=1}^{q}\frac{\partial E_p}{\partial w_1}, \sum_{p=1}^{q}\frac{\partial E_p}{\partial w_2}, \ldots, \sum_{p=1}^{q}\frac{\partial E_p}{\partial w_n}\right).$$

Die Änderung des Gewichtsvektors ist folglich proportional zum negativen Gradienten der Fehlerfunktion. Der Netto-Input der Ausgangsneuronen errechnet sich gemäß der linearen Standard-Propagierungsfunktion zu

$$net_{pj} = \sum_{i=1}^{m} w_{ij} a_{pi},$$

wobei w_{ij} das Gewicht zwischen dem sendenden Neuron *i* und dem empfangenden Neuron *j* ist. Die Besonderheit von mehrlagigen Perzeptren liegt in der Verwendung einer nichtlinearen Sigmoidfunktion

$$a_{pj} = f\left(net_{pj}\right) = f\left(\sum_{i=1}^{m} w_{ij} a_{pi}\right) = \frac{1}{1 + e^{-net_{pj}}}$$

zur Aktivierung der Neuronen. Der Fehler im Trainingsmuster *p* über alle Ausgabeneuronen muß also nach Einsetzen der Aktivierungsfunktion minimal werden, d.h.

$$E_p = \frac{1}{2}\sum_{j=1}^{m}\left(\hat{a}_{pj} - a_{pj}\right)^2 = \frac{1}{2}\sum_{j=1}^{m}\left(\hat{a}_{pj} - f\left(\sum_{i=1}^{m} w_{ij} a_{pi}\right)\right)^2 \to \min.$$

Verbal läßt sich das Lernproblem etwa dadurch beschreiben, daß die Verbindungsgewichte w_{ij} so bestimmt werden sollen, daß die quadratische Gesamtfehlerdifferenz zwischen Soll- und Ist-Ausgabe (wobei letztere eine nichtlineare Funktion des gewichteten Netto-Inputs darstellt) für alle Ausgangsneuronen minimal wird. Mithin ist also der euklidische Abstand zwischen Zielvektor und Ausgangsvektor zu minimieren. Die notwendige Bedingung zur Lösung der Extremwertaufgabe hat zur Konsequenz, daß der Gradient der Fehlerfunktion null werden muß, d.h.

$$\nabla E(W) = \left(\frac{\partial E}{\partial w_{11}}, \frac{\partial E}{\partial w_{12}}, \ldots, \frac{\partial E}{\partial w_{mm}} \right) = 0.$$

Hierbei können die Koeffizienten für sämtliche modifizierbaren Verbindungsgewichte übersichtlicher als Konnektionsmatrix dargestellt werden:

$$W = \begin{pmatrix} w_{11} & \cdots & w_{1j} & \cdots & w_{1m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{i1} & \cdots & w_{ij} & \cdots & w_{im} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mj} & \cdots & w_{mm} \end{pmatrix}.$$

Dem Gradientenabstiegsverfahren liegt nun die Idee zugrunde, die Veränderung der einzelnen Verbindungsgewichte proportional zum negativen Gradienten der Fehlerfunktion vorzunehmen:

$$\begin{aligned} \frac{\partial E_p}{\partial w_{ij}} &= \frac{1}{2} \sum_{l=1}^{m} \frac{\partial E_p}{\partial \left(\hat{a}_{pl} - a_{pl} \right)} \frac{\partial}{\partial net_{pl}} \left(\hat{a}_{pl} - f\left(net_{pl} \right) \right) \frac{\partial net_{pl}}{\partial w_{ij}} \\ &= \sum_{l=1}^{m} \frac{\partial E_p}{\partial a_{pl}} \frac{\partial f\left(net_{pl} \right)}{\partial net_{pl}} \frac{\partial net_{pl}}{\partial w_{ij}}, \end{aligned}$$

wobei wir die Beziehung

$$\frac{\partial E_p}{\partial \left(\hat{a}_{pl} - a_{pl} \right)} = -\frac{\partial E_p}{\partial a_{pl}} = \hat{a}_{pl} - a_{pl}$$

verwendet haben. Wäre die Aktivierungsfunktion linear, so ließe sich jedes mehrschichtige Backpropagation-Netz auf ein zweistufiges Netz ohne verdeckte Schicht reduzieren, da eine hintereinandergeschaltete lineare Propagierungs- und Aktivierungsfunktion zu einer einzigen linearen Funktion zusammengefaßt werden können.

Die zur Erfüllung der Aufgabenstellung benötigte partielle Ableitung der obigen Aktivierungsfunktion lautet

$$\frac{\partial f\left(net_{pl}\right)}{\partial net_{pl}} = \frac{e^{-net_{pl}}}{\left(1+e^{-net_{pl}}\right)^2} = f^2\left(net_{pl}\right)\left(\frac{1}{f\left(net_{pl}\right)}-1\right) = f\left(net_{pl}\right)\left(1-f\left(net_{pl}\right)\right).$$

Leiten wir den Netz-Input

$$net_{pl} = \sum_{k=1}^{m} w_{kl} a_{pk}$$

nach w_{ij} ab, erhalten wir schließlich

$$\begin{aligned}\frac{\partial net_{pl}}{\partial w_{ij}} &= \sum_{k=1}^{m} \frac{\partial}{\partial w_{ij}} w_{kl} a_{pk} = \sum_{k=1}^{m} \frac{\partial w_{kl}}{\partial w_{ij}} \frac{\partial}{\partial w_{kl}} w_{kl} a_{pk} \\ &= \sum_{k=1}^{m} \frac{\partial w_{kl}}{\partial w_{ij}} \frac{\partial}{\partial w_{kl}} \sum_{i=1}^{m} w_{il} a_{pi} = \sum_{k=1}^{m} \frac{\partial w_{kl}}{\partial w_{ij}} \frac{\partial net_{pl}}{\partial w_{kl}}.\end{aligned}$$

Wenn wir nun alle bislang erhaltenen Ausdrücke in den Ausdruck für den Gradienten der obigen Fehlerfunktion einsetzen, lautet dieser am Ende

$$\frac{\partial E_p}{\partial w_{ij}} = -\sum_{l=1}^{m}\left(\hat{a}_{pl} - a_{pl}\right) f\left(net_{pl}\right)\left(1-f\left(net_{pl}\right)\right)\sum_{k=1}^{m} \frac{\partial w_{kl}}{\partial w_{ij}} \frac{\partial net_{pl}}{\partial w_{kl}}.$$

Zusammengefaßt ergibt sich mit

$$\frac{\partial net_{pl}}{\partial w_{kl}} = a_{pk} \quad \text{und} \quad \frac{\partial w_{kl}}{\partial w_{ij}} = 1,$$

und weil $i = k$ und $j = l$ ist, der Ausdruck

$$\frac{\partial E_p}{\partial w_{ij}} = -\left(\hat{a}_{pj} - a_{pj}\right) a_{pj}\left(1-a_{pj}\right) a_{pi}.$$

Als Gewichtsadaption erhalten wir schließlich

$$\Delta w_{ij} = -\gamma \sum_{p=1}^{q} \frac{\partial E_p}{\partial w_{ij}} = \gamma \sum_{p=1}^{q}\left(\hat{a}_{pj} - a_{pj}\right) a_{pj}\left(1-a_{pj}\right) a_{pi}.$$

Die bisherigen allgemeinen Überlegungen lassen sich nun auf eine konkrete Startsituation übertragen, was eine schrittweise Anpassung der Gewichte ermöglicht. Dabei können die Anfangswerte w_{ij}^0 beliebig gewählt werden. Mit dem entsprechenden Netz-Input

$$net_{pj}^0 = \sum_{i=1}^{m} w_{ij}^0 a_{pi}$$

läßt sich direkt der nullte Output

$$a_{pj}^0 = f\left(\sum_{i=1}^{m} w_{ij}^0 a_{pi}\right) = \frac{1}{1+e^{-\sum_{i=1}^{m} w_{ij}^0 a_{pi}}}$$

angeben, und die nullte Gewichtsadaption lautet

$$\Delta w_{ij}^0 = \gamma \sum_{p=1}^{q} \left(\hat{a}_{pj} - a_{pj}^0\right) a_{pj}^0 \left(1 - a_{pj}^0\right) a_{pi}.$$

Damit ergeben sich die Verbindungsgewichte im nächsten Lernschritt aus

$$w_{ij}^1 = w_{ij}^0 + \Delta w_{ij}^0,$$

und mit

$$a_{pj}^1 = f\left(\sum_{i=1}^{m} w_{ij}^1 a_{pi}\right) = \frac{1}{1+e^{-\sum_{i=1}^{m} w_{ij}^1 a_{pi}}}$$

folgt als erstes Differenzgewicht

$$\Delta w_{ij}^1 = \gamma \sum_{p=1}^{q} \left(\hat{a}_{pj} - a_{pj}^1\right) a_{pj}^1 \left(1 - a_{pj}^1\right) a_{pi}.$$

Die zweite Näherung lautet damit

$$w_{ij}^2 = w_{ij}^1 + \Delta w_{ij}^1,$$

und mit

$$a_{pj}^2 = f\left(\sum_{i=1}^{m} w_{ij}^2 a_{pi}\right) = \frac{1}{1+e^{-\sum_{i=1}^{m} w_{ij}^2 a_{pi}}}$$

folgt das zweite Differenzgewicht

$$\Delta w_{ij}^2 = \gamma \sum_{p=1}^{q} \left(\hat{a}_{pj} - a_{pj}^2\right) a_{pj}^2 \left(1 - a_{pj}^2\right) a_{pi}.$$

Analog ergibt sich das dritte Gewicht

$$w_{ij}^3 = w_{ij}^2 + \Delta w_{ij}^2.$$

Allgemein gilt, wenn w_{ij}^n bekannt ist, eine iterative Abfolge folgender Größen:

$$net_{pj}^{n} = \sum_{i=1}^{m} w_{ij}^{n} a_{pi},$$

$$a_{pj}^{n} = f\left(net_{pj}^{n}\right) = \frac{1}{1+e^{-net_{pj}^{n}}},$$

$$\Delta w_{ij}^{n} = \gamma \sum_{p=1}^{q} \left(\hat{a}_{pj} - a_{pj}^{n}\right) a_{pj}^{n} \left(1 - a_{pj}^{n}\right) a_{pi},$$

$$w_{ij}^{n+1} = w_{ij}^{n} + \Delta w_{ij}^{n}.$$

Die Anwendung neuronaler Netzwerke gliedert sich in zwei Phasen, und zwar in eine Lern- und eine Anwendungsphase. In der Lernphase werden dem Netz eine Menge Trainingsgesichter präsentiert, aufgrund derer die Verbindungsgewichte der Neuronen so justiert werden, daß das Netz dem gewünschten Verhalten genügt und die Differenz zwischen dem Ausgabewert und dem gewünschten Sollwert minimal wird. In der Anwendungsphase werden erneut Inputvektoren angeboten, wobei die Ausgabevektoren unter Abruf des erlernten Wissens vom Netz selbständig berechnet werden.

Anhang 1: Programm zur Fehlerberechnung einer Gesichtserkennung

```
% Program Face recognition
clear all

% Positionsmeßdaten 1. Person (Manfred Hiebl)
x(1) = 12.80; y(1) = 02.12;
x(2) = 08.39; y(2) = 04.98;
x(3) = 11.50; y(3) = 07.26;
x(4) = 08.21; y(4) = 07.94;
x(5) = 11.39; y(5) = 08.13;
x(6) = 08.50; y(6) = 10.58;
x(7) = 12.52; y(7) = 10.45;
x(8) = 09.07; y(8) = 12.14;
x(9) = 11.23; y(9) = 11.71;
x(10) = 13.06; y(10) = 14.24;
x(11) = 14.41; y(11) = 11.73;
x(12) = 15.53; y(12) = 12.12;
x(13) = 13.71; y(13) = 10.45;
x(14) = 15.85; y(14) = 10.71;
x(15) = 14.96; y(15) = 08.13;
x(16) = 16.18; y(16) = 08.14;
x(17) = 14.44; y(17) = 07.31;
x(18) = 16.06; y(18) = 04.98;

x(19) = x(1);  y(19) = y(1);
x(20) = x(3);  y(20) = y(3);
x(21) = x(17); y(21) = y(17);
x(22) = x(5);  y(22) = y(5);
x(23) = x(15); y(23) = y(15);
x(24) = x(7);  y(24) = y(7);
x(25) = x(13); y(25) = y(13);
x(26) = x(9);  y(26) = y(9);
x(27) = x(11); y(27) = y(11);

% Positionsmeßdaten 2. Person (Donald Trump)
x(1) = 12.29; y(1) = 03.50;
x(2) = 08.95; y(2) = 05.52;
x(3) = 11.11; y(3) = 07.24;
```



```
x(4) = 08.95; y(4) = 08.33;
x(5) = 10.85; y(5) = 07.92;
x(6) = 09.46; y(6) = 10.33;
x(7) = 11.70; y(7) = 09.53;
x(8) = 09.72; y(8) = 11.13;
x(9) = 11.36; y(9) = 10.61;
x(10) = 12.29; y(10) = 12.22;
x(11) = 12.89; y(11) = 10.58;
x(12) = 14.57; y(12) = 11.16;
x(13) = 12.35; y(13) = 09.53;
x(14) = 14.92; y(14) = 10.37;
x(15) = 13.36; y(15) = 07.86;
x(16) = 15.56; y(16) = 08.33;
x(17) = 13.15; y(17) = 07.26;
x(18) = 15.43; y(18) = 05.52;

x(19) = x(1);  y(19) = y(1);
x(20) = x(3);  y(20) = y(3);
x(21) = x(17); y(21) = y(17);
x(22) = x(5);  y(22) = y(5);
x(23) = x(15); y(23) = y(15);
x(24) = x(7);  y(24) = y(7);
x(25) = x(13); y(25) = y(13);
x(26) = x(9);  y(26) = y(9);
x(27) = x(11); y(27) = y(11);

% Positionsreferenzdaten
x0(1) = 12.19; y0(1) = 03.07;
x0(2) = 09.24; y0(2) = 05.10;
x0(3) = 11.31; y0(3) = 06.61;
x0(4) = 09.18; y0(4) = 07.45;
x0(5) = 11.05; y0(5) = 07.35;
x0(6) = 09.44; y0(6) = 09.74;
x0(7) = 11.83; y0(7) = 09.01;
x0(8) = 09.84; y0(8) = 10.96;
x0(9) = 11.28; y0(9) = 10.25;
x0(10) = 12.39; y0(10) = 12.08;
x0(11) = 13.27; y0(11) = 10.25;
x0(12) = 14.69; y0(12) = 10.96;
x0(13) = 12.96; y0(13) = 09.01;
x0(14) = 15.29; y0(14) = 09.69;
x0(15) = 13.49; y0(15) = 07.35;
x0(16) = 15.35; y0(16) = 07.44;
x0(17) = 13.35; y0(17) = 06.65;
x0(18) = 15.37; y0(18) = 05.10;

x0(19) = x0(1); y0(19) = y0(1);
x0(20) = x0(3); y0(20) = y0(3);
x0(21) = x0(17); y0(21) = y0(17);
x0(22) = x0(5); y0(22) = y0(5);
x0(23) = x0(15); y0(23) = y0(15);
x0(24) = x0(7); y0(24) = y0(7);
x0(25) = x0(13); y0(25) = y0(13);
x0(26) = x0(9); y0(26) = y0(9);
x0(27) = x0(11); y0(27) = y0(11);
x0(28) = x0(10); y0(28) = y0(1);

% Reihe
for i = 1:25
    A0(i)  =  abs((x0(i+1)-x0(i))*(y0(i+2)-y0(i))-(y0(i+1)-y0(i))*(x0(i+2)-
x0(i)));
```



```
    A(i) = abs((x(i+1)-x(i))*(y(i+2)-y(i))-(y(i+1)-y(i))*(x(i+2)-x(i)));
end
X = ['Absolute Soll-Flächenelemente (unkorrigiert)'];
disp(X)

for i = 1:25
    X = ['A0(',num2str(i),') = ',num2str(A0(i))];
    disp(X)
end
disp(' ');
A0(18) = 0;
Flaeche0 = 0;
for i=1:25
    Flaeche0 = A0(i) + Flaeche0;
end
X = ['Referenzfläche(unkorrigiert) = ',num2str(Flaeche0)];
disp(X)
disp(' ');
for i=1:25
    a0(i)=A0(i)/Flaeche0;
end
X = ['Relative Soll-Flächenelemente (unkorrigiert)'];
disp(X)
for i=1:25
    X = ['a_t(1,',num2str(i),') = ',num2str(a0(i))];
    disp(X)
end
disp(' ');

% Flächenausgleich
a0(1) = (a0(1) + a0(17))/2;
a0(2) = (a0(2) + a0(16))/2;
a0(3) = (a0(3) + a0(15))/2;
a0(4) = (a0(4) + a0(14))/2;
a0(5) = (a0(5) + a0(13))/2;
a0(6) = (a0(6) + a0(12))/2;
a0(7) = (a0(7) + a0(11))/2;
a0(8) = (a0(8) + a0(10))/2;

a0(10) = a0(8);
a0(11) = a0(7);
a0(12) = a0(6);
a0(13) = a0(5);
a0(14) = a0(4);
a0(15) = a0(3);
a0(16) = a0(2);
a0(17) = a0(1);
a0(18) = 0;

X = ['Relative Soll-Flächenelemente (ausgeglichen)'];
disp(X)

for i=1:25
    X = ['a_t(1,',num2str(i),') = ',num2str(a0(i))];
    disp(X)
end
disp(' ');
A(18) = 0;
X = ['Absolute Ist-Flächenelemente (unnormiert)'];
disp(X)
```



```
for i = 1:25
    X = ['A(',num2str(i),') = ',num2str(A(i))];
    disp(X)
end
disp(' ');
Flaeche = 0;
for i=1:25
    Flaeche = A(i) + Flaeche;
end
X = ['Ist-Fläche (unnormiert)= ',num2str(Flaeche)];
disp(X)
disp(' ');

for i=1:25
    a(i)=A(i)/Flaeche;
end

A(18) = 0;
X = ['Relative Ist-Flächenelemente(unnormiert)'];
disp(X)
for i=1:25
    X = ['a_i(1,',num2str(i),') = ',num2str(a(i))];
    disp(X)
end
disp(' ');
E = 0;
for i=1:25
    E = (a(i)-a0(i))^2 + E;
end
X = ['Fehler (unkorrigiert)'];
disp(X)
X = ['E = ',num2str(E)];
disp(X)
X = ['Korrekturgrößen'];
disp(X)
h0 = 1/2*sqrt((x0(15)+x0(5)-x0(13)-x0(7))^2+(y0(15)+y0(5)-y0(13)-y0(7))^2);
X = ['h0 = ',num2str(h0)];
disp(X)
h = 1/2*sqrt((x(15)+x(5)-x(13)-x(7))^2+(y(15)+y(5)-y(13)-y(7))^2);
X = ['h = ',num2str(h)];
disp(X)
kappa = h0/h;
X = ['kappa = ',num2str(kappa)];
disp(X)

% Normierte Positionsdaten
for i = 1:27
    x(i) = kappa*x(i);
    y(i) = kappa*y(i);
end
disp(' ');

% Reihe
for i = 1:25
    A(i) = abs((x(i+1)-x(i))*(y(i+2)-y(i))-(y(i+1)-y(i))*(x(i+2)-x(i)));
end
A(18) = 0;
X = ['Absolute Ist-Flächenelemente (normiert)'];
disp(X)
for i = 1:25
    X = ['A(',num2str(i),') = ',num2str(A(i))];
```



```
    disp(X)
end
disp(' ');
Flaeche = 0;
for i=1:25
    Flaeche = A(i) + Flaeche;
end
X = ['Ist-Fläche (normiert) = ',num2str(Flaeche)];
disp(X)
disp(' ');

for i=1:25
    a(i)=A(i)/Flaeche;
end

a(18) = 0;
X = ['Relative Ist-Flächenelemente (normiert)'];
disp(X)
for i=1:25
    X = ['a_i(1,',num2str(i),') = ',num2str(a(i))];
    disp(X)
end

% Flächenausgleich
a(1) = (a(1) + a(17))/2;
a(2) = (a(2) + a(16))/2;
a(3) = (a(3) + a(15))/2;
a(4) = (a(4) + a(14))/2;
a(5) = (a(5) + a(13))/2;
a(6) = (a(6) + a(12))/2;
a(7) = (a(7) + a(11))/2;
a(8) = (a(8) + a(10))/2;

a(10) = a(8);
a(11) = a(7);
a(12) = a(6);
a(13) = a(5);
a(14) = a(4);
a(15) = a(3);
a(16) = a(2);
a(17) = a(1);
a(18) = 0;
disp(' ');
X = ['Relative Ist-Flächenelemente (ausgeglichen)'];
disp(X)

for i=1:25
    X = ['a_i(1,',num2str(i),') = ',num2str(a(i))];
    disp(X)
end

E = 0;
for i=1:25
    E = (a(i)-a0(i))^2 + E;
end
disp(' ');
X = ['Fehler (korrigiert)'];
disp(X)
X = ['E = ',num2str(E)];
disp(X)
```



```
>> facerecognition
Absolute Soll-Flächenelemente (unkorrigiert)
A0(1) = 8.6566
A0(2) = 4.9551
A0(3) = 1.3578
A0(4) = 4.3083
A0(5) = 4.5368
A0(6) = 3.2078
A0(7) = 1.3951
A0(8) = 3.4233
A0(9) = 3.6417
A0(10) = 3.2234
A0(11) = 1.5407
A0(12) = 3.3671
A0(13) = 4.2282
A0(14) = 4.1904
A0(15) = 1.2894
A0(16) = 4.6958
A0(17) = 9.0296
A0(18) = 13.0436
A0(19) = 7.2568
A0(20) = 1.52
A0(21) = 1.708
A0(22) = 4.0504
A0(23) = 1.8758
A0(24) = 1.4012
A0(25) = 2.4676

Referenzfläche(unkorrigiert) = 87.3269

Relative Soll-Flächenelemente (unkorrigiert)
a_t(1,1) = 0.099129
a_t(1,2) = 0.056742
a_t(1,3) = 0.015548
a_t(1,4) = 0.049335
a_t(1,5) = 0.051952
a_t(1,6) = 0.036733
a_t(1,7) = 0.015976
a_t(1,8) = 0.039201
a_t(1,9) = 0.041702
a_t(1,10) = 0.036912
a_t(1,11) = 0.017643
a_t(1,12) = 0.038557
a_t(1,13) = 0.048418
a_t(1,14) = 0.047985
a_t(1,15) = 0.014765
a_t(1,16) = 0.053773
a_t(1,17) = 0.1034
a_t(1,18) = 0
a_t(1,19) = 0.083099
a_t(1,20) = 0.017406
a_t(1,21) = 0.019559
a_t(1,22) = 0.046382
a_t(1,23) = 0.02148
a_t(1,24) = 0.016045
a_t(1,25) = 0.028257

Relative Soll-Flächenelemente (ausgeglichen)
a_t(1,1) = 0.10126
a_t(1,2) = 0.055257
```



```
a_t(1,3) = 0.015157
a_t(1,4) = 0.04866
a_t(1,5) = 0.050185
a_t(1,6) = 0.037645
a_t(1,7) = 0.016809
a_t(1,8) = 0.038056
a_t(1,9) = 0.041702
a_t(1,10) = 0.038056
a_t(1,11) = 0.016809
a_t(1,12) = 0.037645
a_t(1,13) = 0.050185
a_t(1,14) = 0.04866
a_t(1,15) = 0.015157
a_t(1,16) = 0.055257
a_t(1,17) = 0.10126
a_t(1,18) = 0
a_t(1,19) = 0.083099
a_t(1,20) = 0.017406
a_t(1,21) = 0.019559
a_t(1,22) = 0.046382
a_t(1,23) = 0.02148
a_t(1,24) = 0.016045
a_t(1,25) = 0.028257

Absolute Ist-Flächenelemente (unnormiert)
A(1) = 18.9494
A(2) = 9.616
A(3) = 2.7875
A(4) = 8.3401
A(5) = 9.4733
A(6) = 6.3453
A(7) = 2.1669
A(8) = 6.2517
A(9) = 8.0088
A(10) = 3.3377
A(11) = 1.1606
A(12) = 3.1006
A(13) = 5.2898
A(14) = 3.1387
A(15) = 0.9952
A(16) = 5.3988
A(17) = 12.229
A(18) = 0
A(19) = 15.1766
A(20) = 2.5633
A(21) = 2.9274
A(22) = 8.2824
A(23) = 2.7608
A(24) = 1.4994
A(25) = 4.0564

Ist-Fläche (unnormiert)= 143.8557

Relative Ist-Flächenelemente(unnormiert)
a_i(1,1) = 0.13173
a_i(1,2) = 0.066845
a_i(1,3) = 0.019377
a_i(1,4) = 0.057975
a_i(1,5) = 0.065853
a_i(1,6) = 0.044109
a_i(1,7) = 0.015063
```



```
a_i(1,8) = 0.043458
a_i(1,9) = 0.055672
a_i(1,10) = 0.023202
a_i(1,11) = 0.0080678
a_i(1,12) = 0.021554
a_i(1,13) = 0.036772
a_i(1,14) = 0.021818
a_i(1,15) = 0.006918
a_i(1,16) = 0.037529
a_i(1,17) = 0.085009
a_i(1,18) = 0
a_i(1,19) = 0.1055
a_i(1,20) = 0.017819
a_i(1,21) = 0.02035
a_i(1,22) = 0.057574
a_i(1,23) = 0.019191
a_i(1,24) = 0.010423
a_i(1,25) = 0.028198

Fehler (unkorrigiert)
E = 0.0044489
Korrekturgrößen
h0 = 1.6647
h = 2.3208
kappa = 0.7173

Absolute Ist-Flächenelemente (normiert)
A(1) = 9.7499
A(2) = 4.9477
A(3) = 1.4342
A(4) = 4.2912
A(5) = 4.8742
A(6) = 3.2648
A(7) = 1.1149
A(8) = 3.2166
A(9) = 4.1207
A(10) = 1.7173
A(11) = 0.59716
A(12) = 1.5953
A(13) = 2.7217
A(14) = 1.6149
A(15) = 0.51205
A(16) = 2.7778
A(17) = 6.2921
A(18) = 0
A(19) = 7.8087
A(20) = 1.3189
A(21) = 1.5062
A(22) = 4.2615
A(23) = 1.4205
A(24) = 0.77148
A(25) = 2.0871

Ist-Fläche (normiert) = 74.0172

Relative Ist-Flächenelemente (normiert)
a_i(1,1) = 0.13173
a_i(1,2) = 0.066845
a_i(1,3) = 0.019377
a_i(1,4) = 0.057975
a_i(1,5) = 0.065853
```



```
a_i(1,6) = 0.044109
a_i(1,7) = 0.015063
a_i(1,8) = 0.043458
a_i(1,9) = 0.055672
a_i(1,10) = 0.023202
a_i(1,11) = 0.0080678
a_i(1,12) = 0.021554
a_i(1,13) = 0.036772
a_i(1,14) = 0.021818
a_i(1,15) = 0.006918
a_i(1,16) = 0.037529
a_i(1,17) = 0.085009
a_i(1,18) = 0
a_i(1,19) = 0.1055
a_i(1,20) = 0.017819
a_i(1,21) = 0.02035
a_i(1,22) = 0.057574
a_i(1,23) = 0.019191
a_i(1,24) = 0.010423
a_i(1,25) = 0.028198

Relative Ist-Flächenelemente (ausgeglichen)
a_i(1,1) = 0.10837
a_i(1,2) = 0.052187
a_i(1,3) = 0.013148
a_i(1,4) = 0.039897
a_i(1,5) = 0.051312
a_i(1,6) = 0.032831
a_i(1,7) = 0.011565
a_i(1,8) = 0.03333
a_i(1,9) = 0.055672
a_i(1,10) = 0.03333
a_i(1,11) = 0.011565
a_i(1,12) = 0.032831
a_i(1,13) = 0.051312
a_i(1,14) = 0.039897
a_i(1,15) = 0.013148
a_i(1,16) = 0.052187
a_i(1,17) = 0.10837
a_i(1,18) = 0
a_i(1,19) = 0.1055
a_i(1,20) = 0.017819
a_i(1,21) = 0.02035
a_i(1,22) = 0.057574
a_i(1,23) = 0.019191
a_i(1,24) = 0.010423
a_i(1,25) = 0.028198

Fehler (korrigiert)
E = 0.0012898
```

Anhang 2: Programm zur Gewichtsbestimmung bei der Gesichtserkennung

```
% Neural Network Backpropagation Algorithm for Face Recognition
clear all

% Sollwerte
a_t(1,1) = 0.10126;
a_t(1,2) = 0.055257;
a_t(1,3) = 0.015157;
a_t(1,4) = 0.04866;
a_t(1,5) = 0.050185;
```



```
a_t(1,6) = 0.037645;
a_t(1,7) = 0.016809;
a_t(1,8) = 0.038056;
a_t(1,9) = 0.041702;
a_t(1,10) = 0.038056;
a_t(1,11) = 0.016809;
a_t(1,12) = 0.037645;
a_t(1,13) = 0.050185;
a_t(1,14) = 0.04866;
a_t(1,15) = 0.015157;
a_t(1,16) = 0.055257;
a_t(1,17) = 0.10126;
a_t(1,18) = 0;
a_t(1,19) = 0.083099;
a_t(1,20) = 0.017406;
a_t(1,21) = 0.019559;
a_t(1,22) = 0.046382;
a_t(1,23) = 0.02148;
a_t(1,24) = 0.016045;
a_t(1,25) = 0.028257;

% Istwerte 1. Person
a_i(1,1) = 0.10837;
a_i(1,2) = 0.052187;
a_i(1,3) = 0.013148;
a_i(1,4) = 0.039897;
a_i(1,5) = 0.051312;
a_i(1,6) = 0.032831;
a_i(1,7) = 0.011565;
a_i(1,8) = 0.03333;
a_i(1,9) = 0.055672;
a_i(1,10) = 0.03333;
a_i(1,11) = 0.011565;
a_i(1,12) = 0.032831;
a_i(1,13) = 0.051312;
a_i(1,14) = 0.039897;
a_i(1,15) = 0.013148;
a_i(1,16) = 0.052187;
a_i(1,17) = 0.10837;
a_i(1,18) = 0;
a_i(1,19) = 0.1055;
a_i(1,20) = 0.017819;
a_i(1,21) = 0.02035;
a_i(1,22) = 0.057574;
a_i(1,23) = 0.019191;
a_i(1,24) = 0.010423;
a_i(1,25) = 0.028198;

% Istwerte 2. Person
a_i(2,1) = 0.11508;
a_i(2,2) = 0.072449;
a_i(2,3) = 0.013727;
a_i(2,4) = 0.050179;
a_i(2,5) = 0.053765;
a_i(2,6) = 0.024664;
a_i(2,7) = 0.017368;
a_i(2,8) = 0.035517;
a_i(2,9) = 0.028417;
a_i(2,10) = 0.035517;
a_i(2,11) = 0.017368;
a_i(2,12) = 0.024664;
```



```
a_i(2,13) = 0.053765;
a_i(2,14) = 0.050179;
a_i(2,15) = 0.013727;
a_i(2,16) = 0.072449;
a_i(2,17) = 0.11508;
a_i(2,18) = 0;
a_i(2,19) = 0.0873;
a_i(2,20) = 0.015883;
a_i(2,21) = 0.017323;
a_i(2,22) = 0.046679;
a_i(2,23) = 0.012382;
a_i(2,24) = 0.0080077;
a_i(2,25) = 0.01851;

m = 25;
for i = 1:m
    a_t(2,i)= a_t(1,i);
end

gamma = 0.8;
nmax = 2500;
for i = 1:m
    for j = 1:m
        w0(i,j) = 1;
    end
end

q = 2;
for  p = 1:q
    for j = 1:m
        summe = 0;
        for i = 1:m
            summe = summe + w0(i,j)*a_i(p,i);
        end
        net0(p,j) = summe;
        a0(p,j) = 1/(1+exp(-net0(p,j)));
    end
end

for i = 1:m
    for j = 1:m
        summe = 0;
        for p = 1:q
            summe = summe + gamma*(a_t(p,j)-a0(p,j))*(1-a0(p,j))*a_i(p,i);
        end
        Deltaw0(i,j) = summe;
        w(i,j,1) = w0(i,j) + Deltaw0(i,j);
    end
end

for n = 1:nmax
    for j = 1:m
        summe = 0;
        for i = 1:m
            summe = summe + w(i,j,n)*a_i(p,i);
        end
        net(p,j,n) = summe;
        a(p,j,n) = 1/(1+exp(-net(p,j,n)));
        for i = 1:m
            summe = 0;
```



```
            for p = 1:q
                summe     =     summe     +     gamma*(a_t(p,j)-a(p,j,n))*(1-
a(p,j,n))*a_i(p,i);
            end
            Deltaw(i,j,n) = summe;
            w(i,j,n+1) = w(i,j,n) + Deltaw(i,j,n);
        end
    end
end

i = 13;
j = 13;
for n = 1:nmax
    f(n) = w(i,j,n);
end

X        =        ['w(',num2str(i),',',num2str(j),',',num2str(n),')        =
',num2str(w(i,j,n))];
disp(X)

figure(1)
% plot(n,f)
stairs(f)
grid on
xlim([0 nmax])
ylabel('$w_{ij}^n$','interpreter','latex')
xlabel('$n$','interpreter','latex')
title('Gewichte w_{13,13}')

for n = 1:nmax
    g(n) = Deltaw(i,j,n);
end

figure(2)
stairs(g)
grid on
xlim([0 nmax])
ylabel('$\Delta w_{ij}^n$','interpreter','latex')
xlabel('$n$','interpreter','latex')
title('Gewichtszuwächse w_{13,13}')

for i = 1:m
    for j = 1:m
        u(i,j) = w(i,j,nmax);
    end
end

figure(3)
surf(u)
zlabel('$w_{ij}$','interpreter','latex')
xlabel('$i$','interpreter','latex')
ylabel('$j$','interpreter','latex')
```

>> NNfacerecognition
w(13,19,2500) = -1.1617

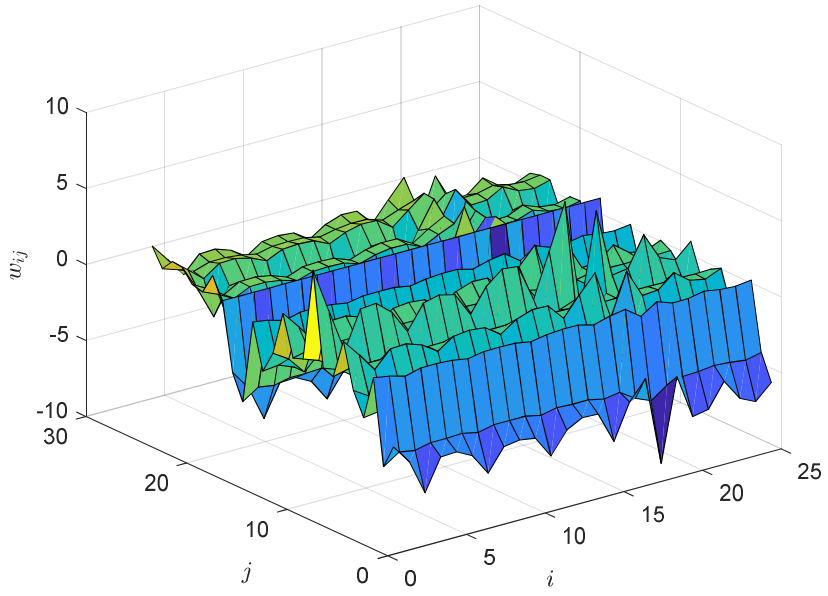

Abbildung 7. Matrix der Verbindungsgewichte

Abb. 7 zeigt die Konnektionsmatrix, die in zwei Dimensionen noch anschaulich dargestellt werden kann. In Abb. 8 erkennen wir, wie sich die Gewichte im Lauf von 2500 Schritten ihrem Grenzwert annähern, und die letzte Abb. 9 zeigt, daß auch die Gewichtszuwächse irgendwann nicht mehr wachsen, sondern stationär werden.

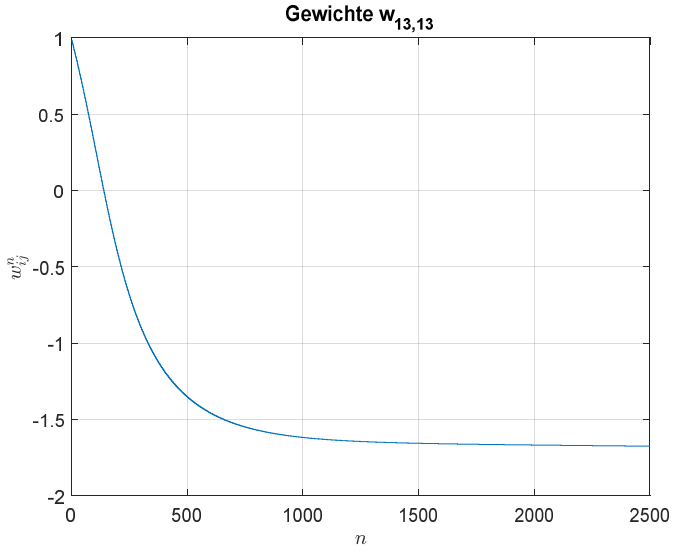

Abbildung 8. Iterative Gewichtsapproximation für i=j=13

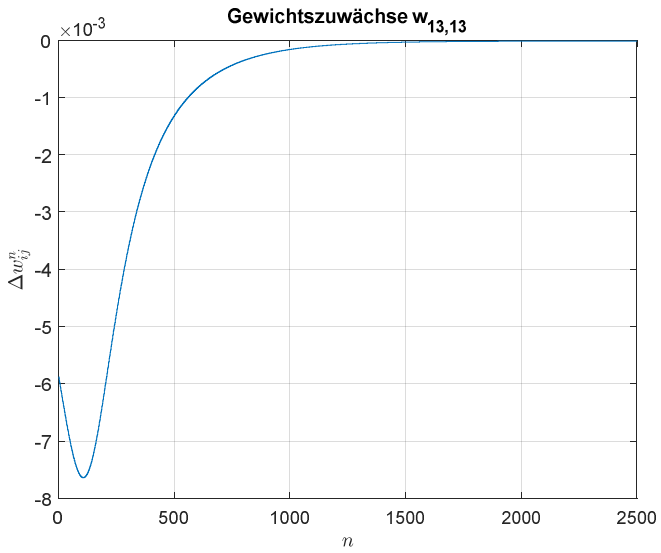

Abbildung 9. Iterativer Verlauf der Gewichtszuwächse für i=j=13

