Klassifikation von Gesichtern mit einem künstlichen neuronalen Netz

Voraussetzungen:

Was wird benötigt?

- **MemBrain** (nur Windows, kostenlos unter membrain-nn.de)
- Texteditor wie **Notepad++** (zum Erstellen reiner Textdateien)
- **Gimp** (open-source unter gimp.org)
- **Yale-Face-Datensatz** (http://vision.ucsd.edu/content/yale-face-database)

Die folgende Darstellung beschränkt sich lediglich auf die Modellierung eines neuronalen Netzes zur Gesichterklassifikation. Sinnvoll ist es, zuvor in das Thema einzuführen und auch schon mit MemBrain gearbeitet zu haben.

Nichtsdestotrotz ist es möglich, die Lehrer- und Schüleranleitung auch ohne jegliche Vorkenntnisse schrittweise durchzuarbeiten.

Der hier dargestellte Prozess der Netzerstellung in MemBrain ist auch unter dem Video
https://youtu.be/KzdVPi4RNMg
*in der **ursprünglichen** Form zu finden.*

HINWEIS: Dies ist ein Update der Originalpublikation. Der in der Originalpublikation verwendete Datensatz ist nicht mehr öffentlich zugänglich, daher wird auf den Yale Face Datensatz[1] zurückgegriffen, der sich jedoch nicht so gut für den gesamten Prozess eignet und einige Anpassungen erfordert. Der Originaldatensatz war auf den Augenabstand normiert und die Gesichter waren in nahezu biometrischer Manier immer in gleicher Perspektive abgelichtet. Dies erlaubte die in der Unterrichtsreihe verwendete Klassifikation „vom alten Typ“ - quasi so wie es ab den 1990er Jahren und vor der Entwicklung der Convolutional Neural Networks[2] gemacht werden musste: ein möglichst normalisiertes Objekt (in diesem Fall Gesicht) ausschneiden, zentrieren, Merkmale bestimmen und dann trainieren und testen.

WICHTIG: Bei der Verwendung des neuen Datensatzes sollten daher die in rot gehaltenen Hinweise im Text beachtet werden.

[1] Science on Stage Deutschland e.V. hat die ausdrückliche Genehmigung von Dr. David Kriegman, University of Califormia San Diego erhalten, dass dieser Datensatz im schulischen Rahmen zu Übungszwecken genutzt werden darf.
[2] Bei neueren Verfahren wie Convolutional Neural Networks spielt die Größe, Rotation und Lage eines Objektes in einem Bild keine Rolle mehr. Diese Verfahren wurden in der Unterrichtsreihe jedoch bewusst nicht verwendet.

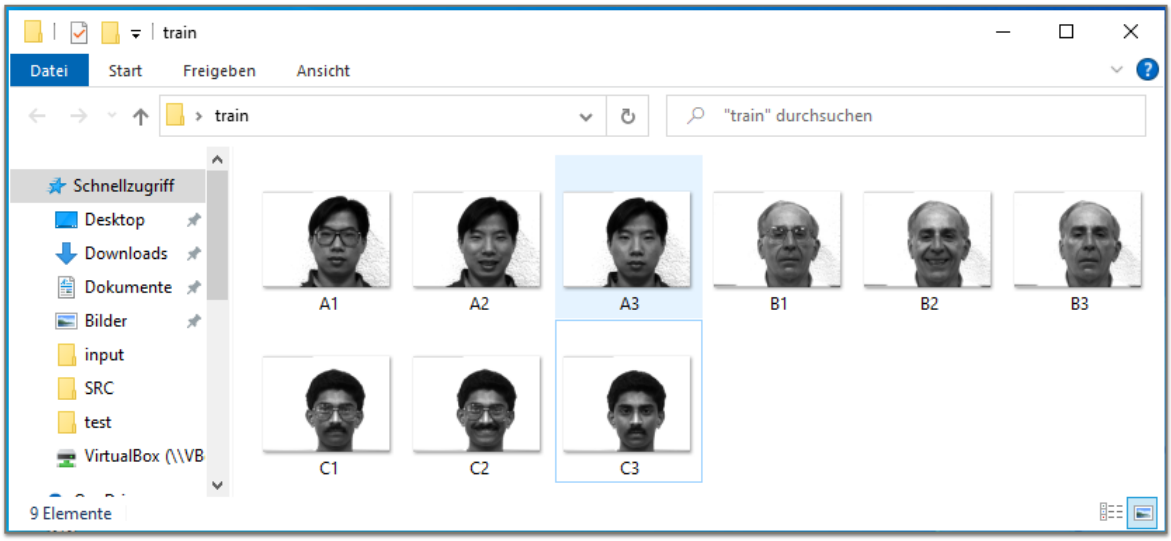
Gesichtserkennung:

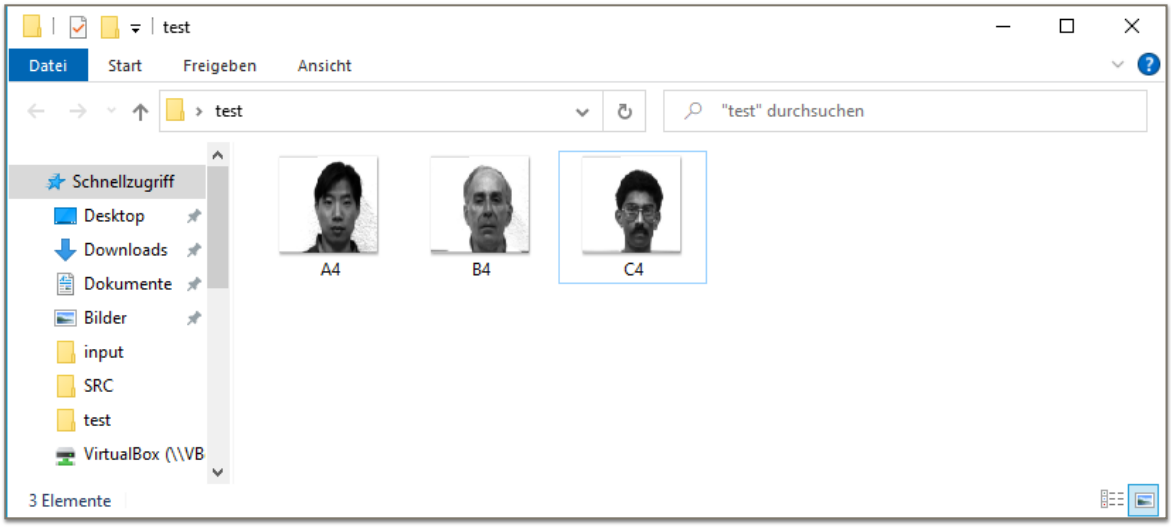
Übersicht:

- **Lehrkraft verteilt Bilder an Schülergruppen**
- **SuS entwickeln Gesichtsmodell**
- **SuS erheben die Gesichtsdaten für die CSV-Trainingsdatei mit GIMP**
- **SuS erstellen in MemBrain das Künstliche Neuronale Netz (KNN)**
- **SuS laden die CSV-Trainingsdatei und trainieren das KNN**
- **SuS erheben die Gesichtsdaten für die CSV-Testdatei mit GIMP**
- **SuS laden die CSV-Testdatei und testen das KNN**

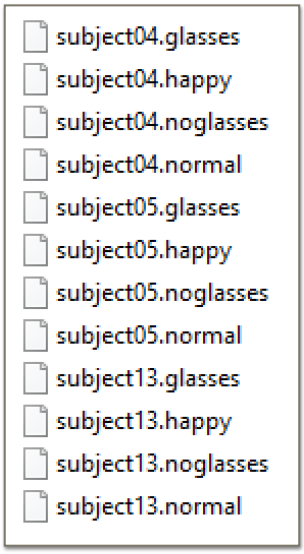
Vorbereitung

Die Lehrkraft verteilt pro Schülergruppe aus dem Yale Face-Datensatz einen Ordner mit Trainingsdaten und einen Ordner mit Testdaten. Die Trainingsdaten könnten zum Beispiel aus je drei Bildern von je drei Personen bestehen, also insgesamt neun Bilder enthalten. Die Testdaten könnten zum Beispiel aus je einem Bild von je drei Personen bestehen, also insgesamt drei Bilder enthalten. Die nachfolgenden Abbildungen visualisieren dies für den Ordner Trainingsdaten und den Ordner Testdaten:





Tipp: Für den Erkenntnisgewinn reicht diese Beschränkung auf wenige Bilder vermutlich aus. Die Bilder der nebenstehenden Liste führten dabei zu einigermaßen guten Ergebnissen. Man könnte die Datenmenge jedoch auch problemlos erhöhen, indem man z.B. jeder Gruppe eigene (also andere insgesamt 12) Bilder gibt und am Ende neben den gruppeneigenen Trainings- und Testtabellen eine gemeinsame Trainings- und Testdatei aus möglichst vielen unterschiedlichen Personen erstellt. Hier muss man dann jedoch später bei der Kodierung der Ausgabe aufpassen, so dass man auch wirklich mit z.B. 8 Klassen für 8 verschiedene Personen arbeitet. Ein Ansatz hierzu wird weiter unten als Alternative vorgestellt.

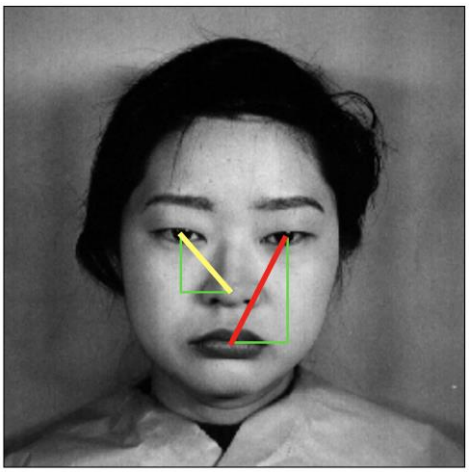


*Angesichts der im Vergleich zum Originaldatensatz „schwierigeren" Bilder wird hiervon jedoch abgeraten. Am besten funktioniert die Klassifikation mit den Bildern der Personen **4, 5 und 13** aus dem Yale Face-Datensatz - diese unterscheiden sich recht gut, was nämlich bei ungünstig gewählten Merkmalen bei allen anderen Personen nicht so funktioniert. Welche vier Fotos einer Person gewählt werden ist weniger wichtig: gut funktionieren z.B. die Bedingungen glasses, happy, noglasses und normal. Die Bilder sind ganz normale gif-Dateien, die sich in GIMP öffnen lassen, auch wenn sie keine solche Dateiendung haben. Die Lehrkraft könnte die Bilder bereits vorsortieren und ggf. umbenennen.*

Gesichtsmodell

Für die Gesichtserkennung muss zunächst ein Gesichtsmodell entwickelt werden. Dazu sollen die SuS anhand der Fotos aus dem Yale Face-Datensatz ein **eigenes** Gesichts-Modell (mit nur **vier** Merkmalen) entwickeln. Prinzipiell können die SuS auch eigene Fotos verwenden, es wird jedoch empfohlen, auf den o.g. Datensatz zurückzugreifen.

Ein mögliches Gesichtsmodell könnte z.B. darin bestehen, folgende vier Merkmale zu wählen:



- Distanz rechte Pupille zu Nasenspitze (in Pixeln)
- Winkel rechte Pupille zu Nasenspitze (in °)
- Distanz linke Pupille zu Mundmitte (in Pixeln)
- Winkel linke Pupille zu Mundmitte (in °)

Tipp: Den Augen- oder Ohrenabstand zu nehmen, ist nicht sinnvoll, da dieser bei allen Personen sehr ähnlich ist. Winkel zwischen Augen, Nase, Ohren, Mund, Kinn usw. kombiniert mit Strecken oder Streckenverhältnissen führen hingegen zu besseren Klassifikationsleistungen.

*Auch hier könnte man mit mehr Merkmalen arbeiten wie in ‚**echten**' Anwendungen zur Gesichterklassifikation, die Beschränkung auf vier Merkmale dient nur der Zeitersparnis. Das Prinzip bleibt dasselbe.*

Datenerhebung in GIMP

Die SuS öffnen GIMP und verfahren für alle Fotos aus ihrem Trainingsordner wie folgt:

Sie öffnen ein Foto mit **GIMP** (ggf. eignen sich auch andere Programme dazu), indem sie es z.B. per Drag&Drop vom Ordner in das Programm ziehen oder es über das Dateimenü öffnen. Danach **zoomen** sie soweit in das Bild hinein, dass es möglichst groß auf dem Bildschirm zu erkennen ist. Mit dem Werkzeug **Maßband** kann man nun ganz einfach **Distanzen** (in Pixel) und **Winkel** (in °) messen. Dazu das Maßband-Werkzeug auswählen und auf die beiden zu trackenden Punkte klicken. GIMP zeigt in einem kleinen Infofeld die gewünschten Informationen an, die die SuS dann einfach in eine Textdatei schreiben können. Wie das am krisensichersten geht wird im Folgenden beschrieben.

Erstellen einer für MemBrain geeigneten CSV-Datei mit den Trainingsdaten

Es ist möglich, die Werte, die die SuS in GIMP erheben, direkt in eine Excel-Tabelle zu schreiben und diese später als CSV-Datei zu speichern. Am besten entscheidet man sich jedoch für den krisensichersten Weg und lässt die SuS z.B. mit dem Programm Notepad oder Notepad++ o.ä. eine reine Textdatei erstellen, die man unter der Dateiendung **.csv** speichert (Wichtig: nicht .txt). In diese schreiben die SuS nun für jedes bearbeitete Bild genau eine Zeile hinein, so dass sich eine Datei ergibt, die wie das folgende Beispiel aufgebaut ist:

```
M1;M2;M3;M4;Y
0.55;0.43;0.72;0.26;0
0.57;0.43;0.72;0.24;0
0.56;0.43;0.74;0.23;0
0.50;0.41;0.69;0.21;0.5
0.53;0.41;0.72;0.22;0.5
0.50;0.43;0.71;0.24;0.5
0.50;0.45;0.60;0.26;1
0.51;0.45;0.58;0.27;1
0.47;0.45;0.62;0.25;1
```

Tipp: Kopieren Sie zu Übungszwecken ruhig die abgebildeten Werte und speichern Sie diese in eine Datei, z.B. unter dem Namen train.csv. Die Daten enthalten die Merkmale der Personen 4, 5 und 13 unter den Bedingungen glasses, happy und noglasses. Es sind nur die Trainingsdateien.

Dargestellt sind hier die vier Merkmale (hier mit zwei Nachkommastellen) jeweils gefolgt durch ein Semikolon. Ganz rechts wird dann noch die Nummer der Person eingetragen, zu der das Foto gehört. An dieser Stelle **müssen** jedoch **drei wichtige Vereinbarungen** eingehalten werden:

1. **Die erste Zeile der CSV-Datei muss die Namen der später in MemBrain angelegten Neuronen** als Spaltenüberschrift enthalten. Es wird geraten, diese einfach M1, M2, M3 und M4 zu nennen. Als ‚letzte Spaltenüberschrift' darf der Name des Ausgabeneurons nicht fehlen. Am besten nimmt man hier den Namen Y.

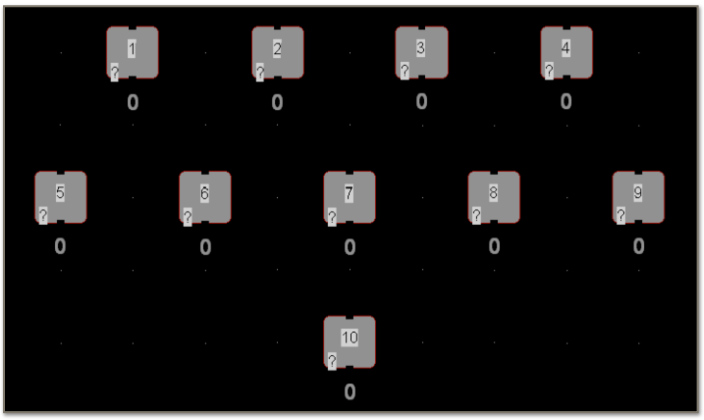
2. **Der Eingabebereich muss im Bereich [0..1] liegen**. Rohwerte müssen vor der Verarbeitung mit neuronalen Netzen auf den Wertebereich der Aktivierungsfunktionen normiert werden. Damit man sich hier aus Gründen der didaktischen Reduktion jedoch nicht mit Normierungsverfahren herumschlagen muss, reicht es aus, dass die SuS einfach eine 0 gefolgt von einem Punkt vor ihren Wert schreiben. Haben sie also eine 81 als Wert in GIMP gemessen, wird 0.81 in die CSV-Datei eingetragen. Dies gilt für die ersten vier Werte in jeder Zeile.

3. Der letzte Wert in jeder Zeile enthält das so genannte **Klassenlabel**. Hier muss also kodiert werden, um **welche Person** es sich bei den vier Merkmalen handelt. Da auch hier der Wertebereich [0..1] gewählt werden muss, bietet es sich an, die erste Person mit 0 zu kodieren, die zweite Person mit 0,5 und die dritte Person mit 1.

Erstellen des KNN in Membrain

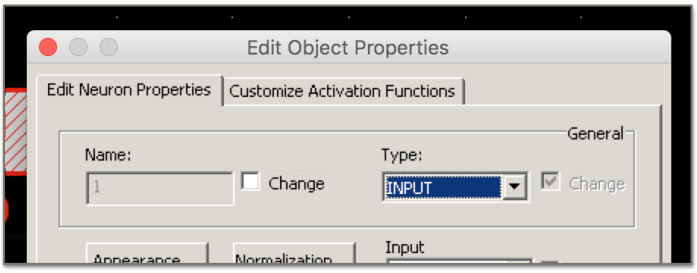
Damit die Klassifikation erfolgreich verläuft, müssen zunächst einige Dinge vorbereitet werden. Dies betrifft die Modellierung des Netzes, die richtigen Einstellungen und das Laden der Trainingsdaten.

1. Zunächst soll die Modellierung erfolgen. Mit dem Insert new neurons-Symbol  können nun per Klick Neuronen auf dem Bildschirm verteilt werden. Wir benötigen vier Eingabeneuronen und ein Ausgabeneuron, wenn wir vier Merkmale auf ein Klassenlabel Y abbilden wollen. Dazwischen können sich beliebig viele Schichten von so genannten versteckten Neuronen befinden. Für unsere Zwecke reicht ein kleines Netz mit drei Schichten (Eingabeschicht, verdeckte Schicht und Ausgabeschicht) und ein paar zusätzlichen Neuronen in der verdeckten Schicht mehr als aus. Die Topologie (also quasi das Aussehen des Netzes) ist dabei Geschmacks- und Erfahrungssache und sollte ruhig in den verschiedenen SuS-



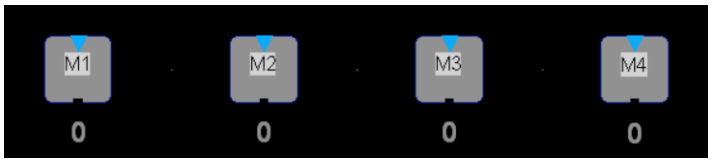
Gruppen unterschiedlich sein.

2. Da die Neuronen unterschiedliche Aufgaben haben (Eingabe, verdeckt, Ausgabe), müssen diese noch festgelegt werden. Dazu kann man in MemBrain bei gedrückter Maustaste gleich mehrere Neuronen auswählen und dann auf eines mit einem Doppelklick klicken, um die Eigenschaften für alle ausgewählten Neurone gleichzeitig festzulegen. In der oberen Reihe muss dazu im aufspringenden Menü unter Type der Eintrag INPUT gewählt werden.

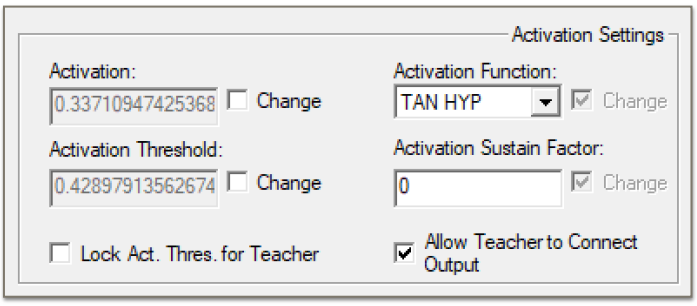


Im Anschluss müssen die Neuronen noch dieselben Namen wie in der CSV-Datei angegeben haben. Dazu klickt man nacheinander auf jedes Eingabeneuron und ändert den Namen des Neurons unter Name entsprechend ab. Also M1, M2, M3 und M4.

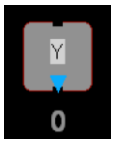
Die Eingangsschicht sollte dann wie folgt aussehen:



Die Neuronen in der verdeckten Schicht müssen auch bearbeitet werden. Man kann einfach alle gleichzeitig auswählen und auf eines doppelklicken. Es öffnet sich dann ein Eigenschaftenfenster, in dem alle verdeckten Neuronen gleichzeitig bearbeitet werden können. Hier muss unter Activation Settings in der Mitte beim Punkt Activation Function die Aktivierungsfunktion TAN HYP (Tangens hyperbolicus) ausgewählt werden. Diese Funktion skaliert die Gewichte zwischen den Neuronen auf den Wertebereich von [-1 .. 1]. So können verdeckte Neuronen quasi sowohl hemmend als auch aktivierend wirken.

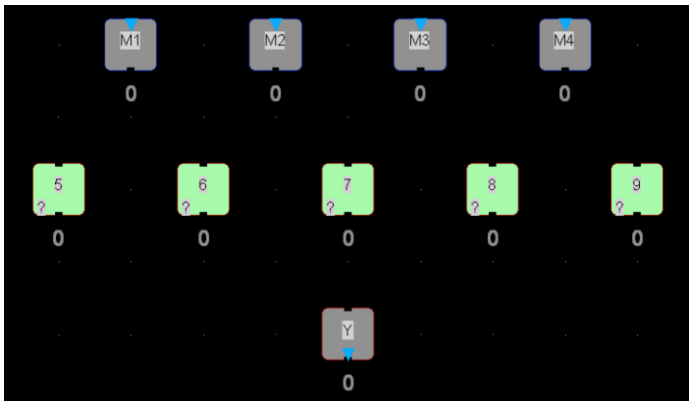


Das einzelne Ausgabeneuron wird per Doppelklick ausgewählt, danach wird im Feld Name der Name Y eingetragen und es wird unter Type als OUTPUT deklariert. Das Ausgabeneuron sieht nun also wie folgt aus (die kleinen blauen Pfeile zeigen übrigens den Verwendungszweck an):

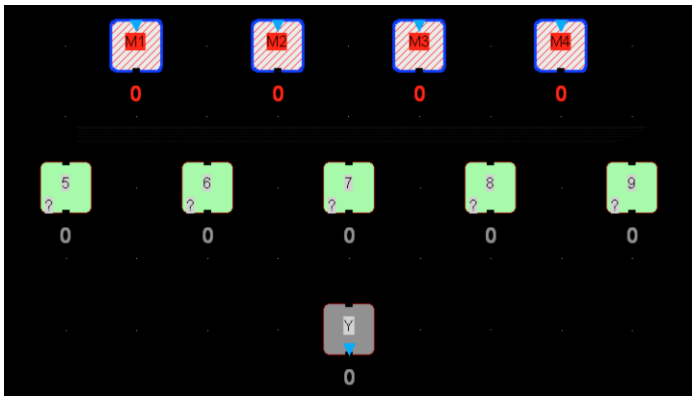


3. Damit auch wirklich ein Netz entsteht, müssen die Neuronen miteinander verbunden werden. Dazu bietet MemBrain das Konzept der **Extraauswahl**. Mit dem Symbol Add to Extra Selection kann man gleich mehrere Neuronen in die Extraauswahl aufnehmen. Sinnvollerweise markiert man dazu

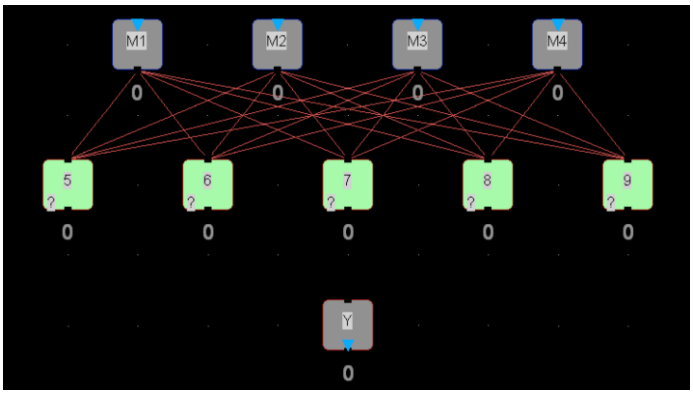
die Neuronen der verdeckten Schicht und klickt auf das abgebildete Symbol. Die Neuronen erscheinen daraufhin grün.



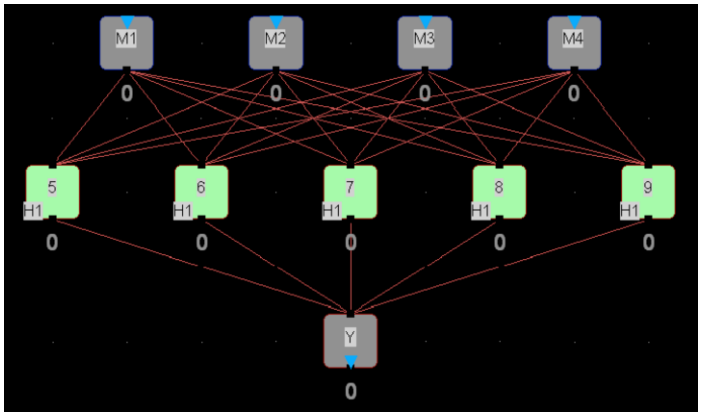
Nun ist es möglich, diese alle auf einmal mit der kompletten oberen Eingangsschicht zu verbinden. Dazu wird jetzt die Eingangsschicht ausgewählt (mit gedrückter Maustaste alle Neuronen der Eingangsschicht auswählen) …



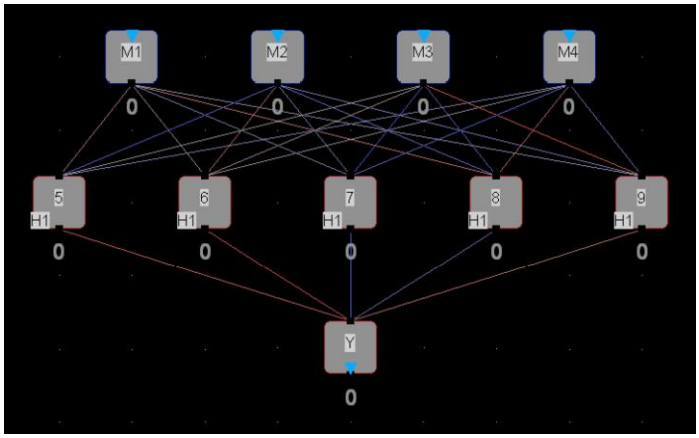
… und danach dann auf das Symbol Connect TO Extra Selection klicken. Nun werden die beiden Schichten automatisch verbunden und man spart sich Bearbeitungszeit. Das Ergebnis sieht dann wie folgt aus:



Danach reicht ein Klick auf das Ausgabeneuron, um danach mit einem Klick auf das Symbol Connect FROM Extra Selection die verdeckte Schicht vollständig mit dem Ausgabeneuron zu verbinden.



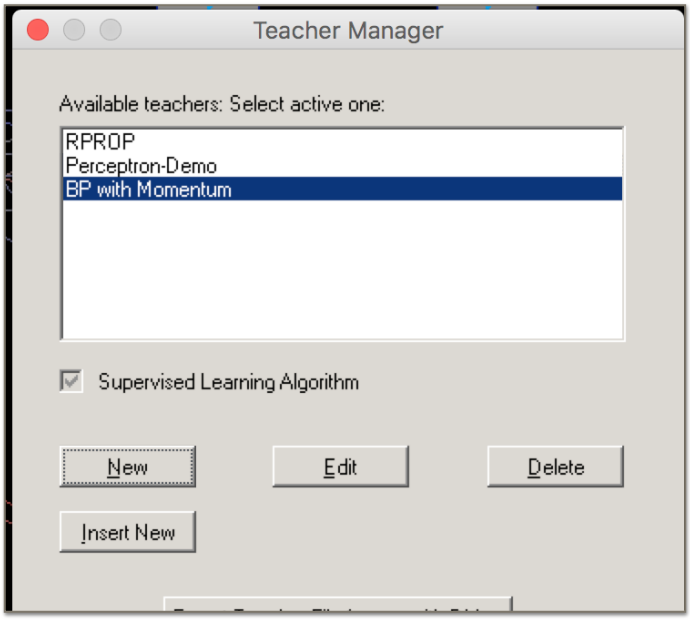
Es bleibt, zuletzt noch die Extraauswahl wieder rückgängig zu machen. Dies geht mit einem Klick auf das Symbol Clear Extra Selection.



Das Netz ist nun bereit für die Gesichtserkennung.

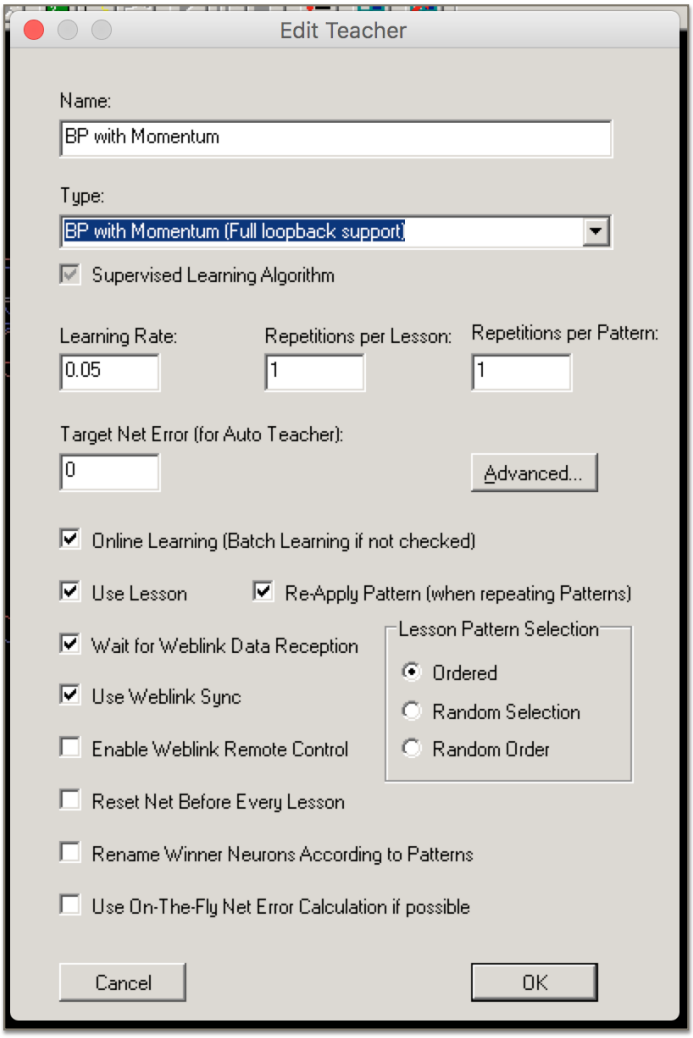
4. Der Teacher Manager, der im Menü Teach zu finden ist, ist Steuerzentrale für unterschiedliche Lernalgorithmen. Von ganz einfachen Lernverfahren bis zu komplizierteren Varianten ist hier zunächst die richtige Auswahl zu treffen, da das richtige Lernverfahren Grundvoraussetzung für den Erfolg des KNNs ist. Würde man bei der Einführung des Themas noch auf Standard Backpropagation (Std. BP) setzen und alle Werte so wählen, dass sie im Kern den Berechnungen der Hebb'schen Lernregel folge, sollte man bei der Gesichtserkennung den folgenden Anweisungen folgen:

Klicken Sie im Menü Teach auf den Eintrag Teacher Manager. Es öffnet sich folgender Dialog (mit eventuell weniger Einträgen bei einer Neuinstallation des Programms):



Klicken Sie nun auf New, um eine neue Lernregel zu erstellen. Wählen Sie BP with Momentum (Full loopback support) oben aus der Liste des Dialogfensters.

Stellen Sie danach die Parameter, wenn nötig, so ein wie auf dem nebenstehenden Bild gezeigt:



Mit einem Klick auf OK gelangen Sie zurück, wählen dann noch aus der Liste das neu erstellte Verfahren aus und bestätigen die Auswahl abermals mit OK. Ab jetzt wird das KNN nun immer diesen Lernalgorithmus verwenden.

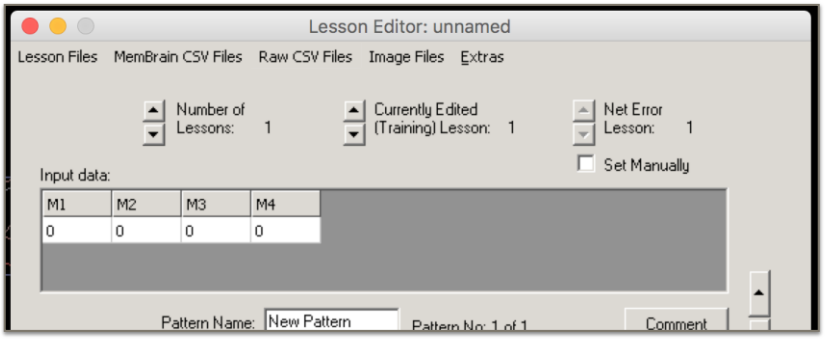
5. Mit einem Klick auf das Symbol Randomize Net initialisieren Sie das Netz mit zufälligen Synapsengewichten, was ein wichtiger Vorgang ist. Denn sollte einmal ein Training zu sehr schlechten Resultaten führen, kann dies an ungünstigen Startwerten in den Synapsengewichten liegen. Mit der Randomisierung kann man es dann erneut mit anderen Startwerten probieren. Auch jedes Mal, wenn das Netz wieder ganz neu trainiert werden soll, muss es wieder randomisiert werden.

Hinweis: Wenn man aufpasst, dass man das Modell nicht mit dem Original verwechselt, kann man hier argumentieren, dass Menschen ebenso individuelle ‚Startwerte' besitzen und das Lernen des Gleichen trotzdem funktionieren kann.

Die Farbgebung der Verbindungen zwischen den Netzen repräsentiert im Übrigen wie bei den Neuronen einen numerischen Wert. Die Erstellung ist nun abgeschlossen und das Netz ist bereit, trainiert zu werden.

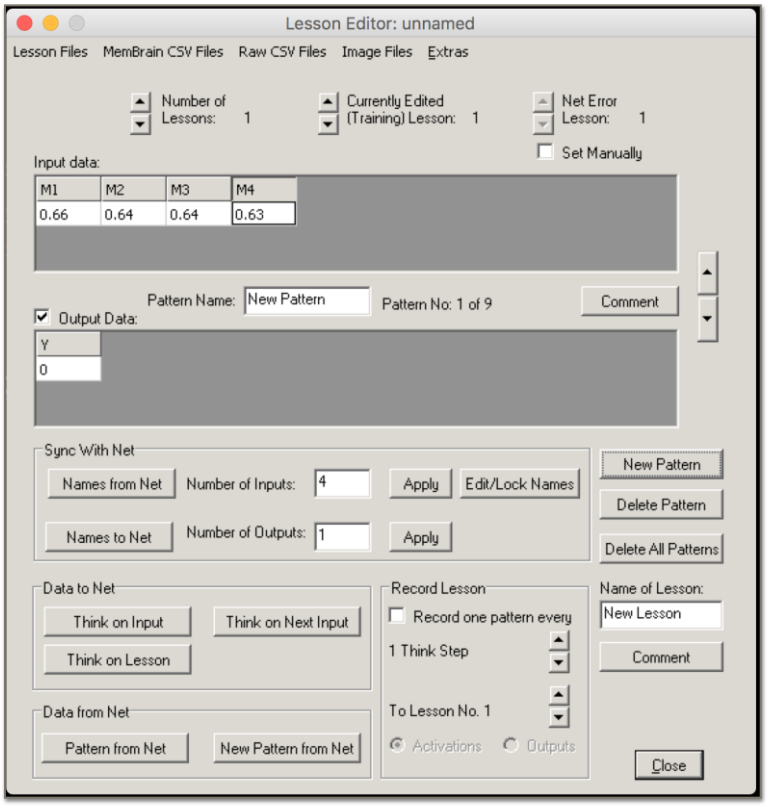
Trainingsdaten in MemBrain laden

Mit einem Klick auf das Symbol Show Lession Editor wird ein Dialogfenster geöffnet, mit dem man u.a. CSV-Dateien bequem importieren kann.



Im Menü Raw CSV Files ist unter dem Menüpunkt Import Current Lesson (Raw CSV) … die Funktionalität versteckt, die Trainingsdaten-Datei zu laden.

Nachdem Sie die richtige Datei ausgewählt haben, werden die Muster automatisch geladen und es sollte auch angezeigt werden, dass nun neun Muster verfügbar sind, durch die man mit den beiden Pfeil-Buttons navigieren kann, evtl. auch, um eine finale Kontrolle vorzunehmen. Ist alles in Ordnung, kann der Dialog mit einem Klick auf Close geschlossen werden.



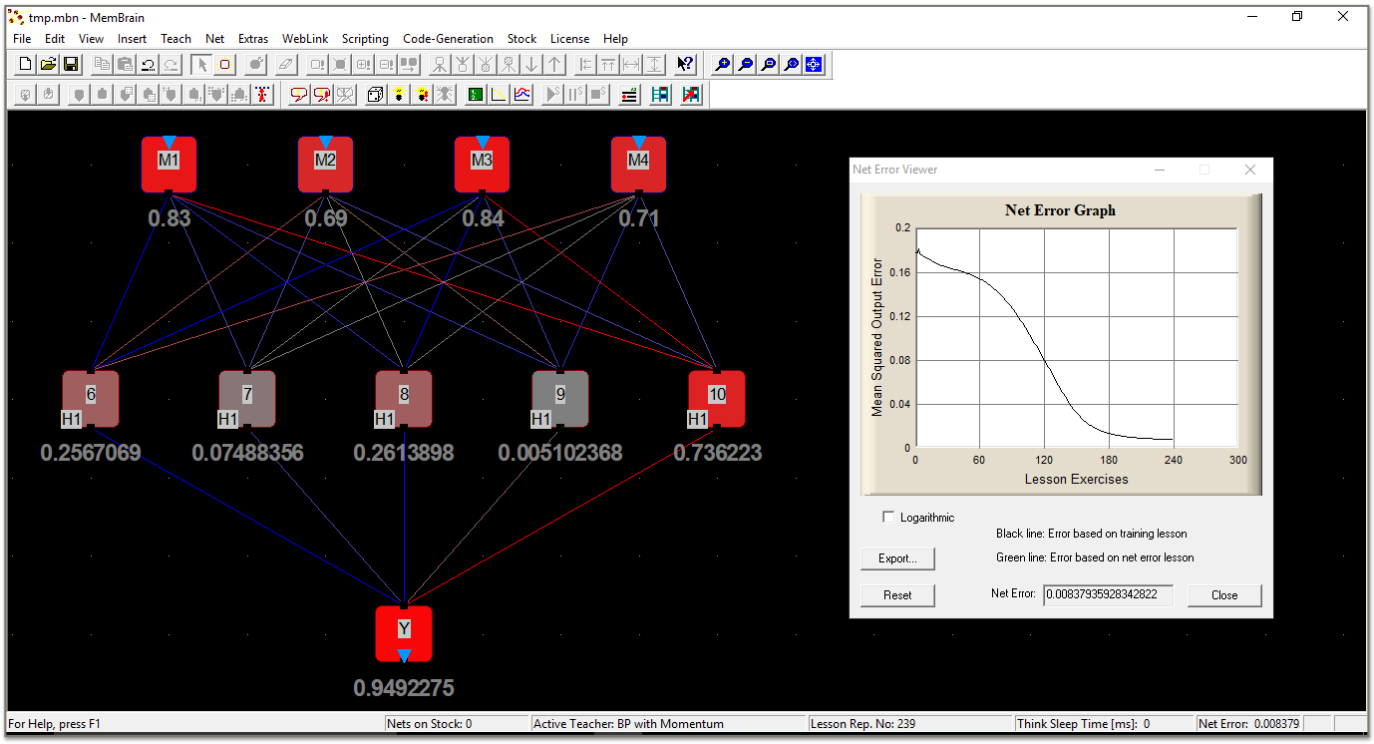
Das Training

Bevor das Training startet, sollte unbedingt noch mit einem Klick auf das Symbol Show Net Error Viewer das Fenster für den Fehler des Netzes geöffnet werden. Dieses kann währenddessen geöffnet bleiben und zeigt sehr schön, wie der Fehler des Netzes während des Trainings minimiert wird bis das Netz alle Gesichter ausreichend gut gelernt hat.

Mit einem Klick auf das Symbol Start Teacher kann das Training nun endlich gestartet werden.

Jedoch wird bei der geringen Menge der Daten der Fehler des Netzes im Error Viewer sehr schnell sinken, weswegen ein baldiger Klick auf das Symbol Stop Teacher erfolgen sollte, sobald die Fehlerkurve erkennbar ihren tiefsten Punkt erreicht.

Hinweis: Zur Demonstration ist die Datenmenge bei der Netzgröße in Ordnung. Für wirkliche Anwendungen, wie sie z.B. Google bei der semantischen Bildersuche verwendet, werden die dortigen (wesentlich größeren und komplexeren) neuronalen Netze mit einer Anzahl an Bildern in einer Größenordnung von weit über vielen Milliarden Bildern trainiert. Problematisch ist nämlich bei zu kleinen Datensätzen, dass es zum so genannten Overtraining kommt - das Netz im Prinzip also nur die wenigen Muster „auswendig lernt“, statt ein generelles ‚Verfahren’ der Wiedererkennung implizit zu entwickeln.



Das KNN ist nun trainiert und bereit zum Einsatz mit dem Test-Datensatz.

Testen des KNNs

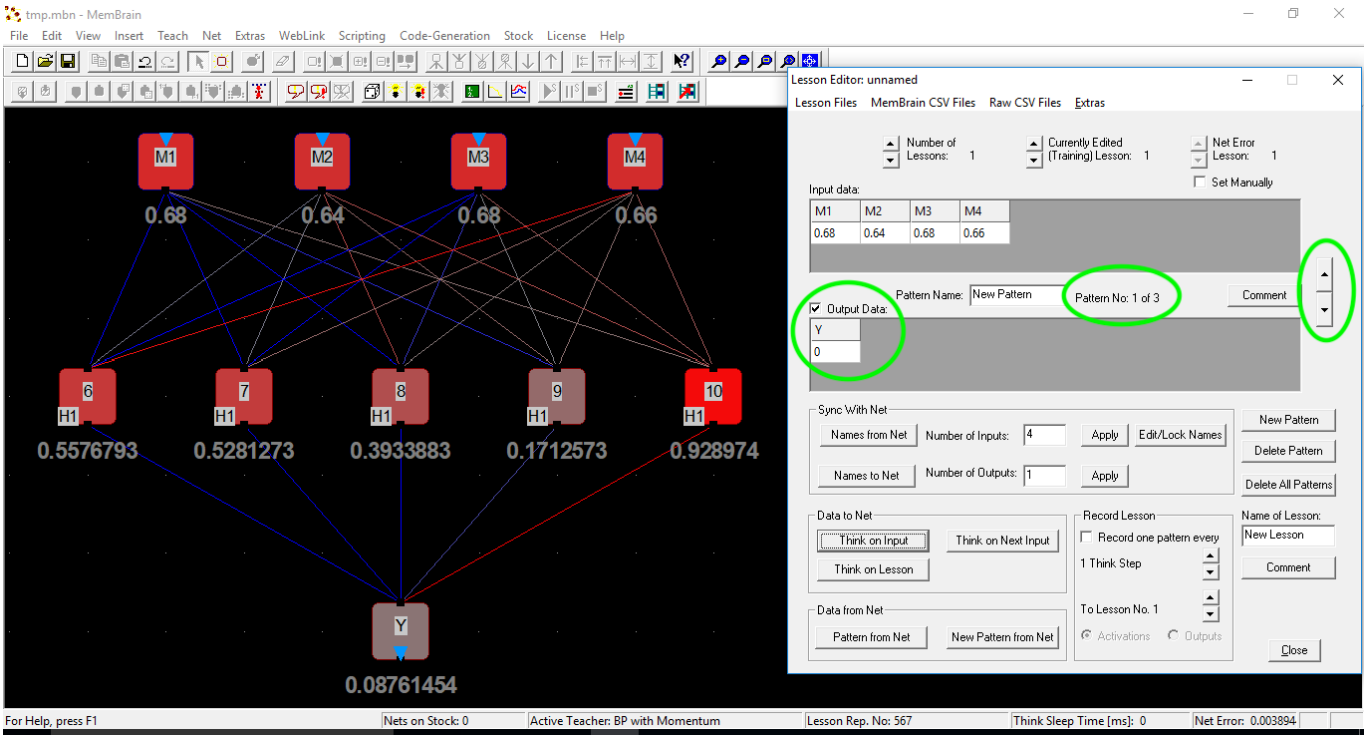
1. Zum Testen des Netzes erstellen die SuS zunächst eine **neue** CSV-Datei, die sie z.B. unter dem Namen test.csv abspeichern. Diese Datei muss exakt gleich aufgebaut sein wie die Trainingsdaten-Datei. Wieder ermitteln sie (dieses Mal nur für drei Fotos) die vier exakt gleichen Merkmale und tragen sie dann entsprechend in die CSV-Datei ein.

Die fertige CSV-Datei könnte z.B. wie folgt aussehen:

```
M1;M2;M3;M4;Y
0.57;0.42;0.74;0.23;0
0.50;0.41;0.71;0.22;0.5
0.50;0.45;0.60;0.26;1
```

2. In MemBrain gilt es nun, die Testdaten in den Lesson Editor zu laden. Dieser wird mit einem Klick auf das Symbol Show Lession Editor geöffnet. Auf die gleiche Art und Weise wie eine Trainingsdatei importiert wurde, wird nun die CSV-Testdatei geladen. Sie ersetzt nun damit die bisher neun vorhandenen Muster durch die drei Muster, die sich in der Datei befinden. Auch hier bietet es sich an, nochmals zu kontrollieren, ob alle Werte richtig übernommen wurden. Ist das der Fall, kann der Test direkt im Lession Editor gestartet werden.

3. Das Dialogfenster sollte am besten so auf dem Bildschirm platziert werden, dass neben ihm auch das KNN sichtbar ist. Für das Testen sind dann alle in der folgenden Abbildung grün markierten Teile des



Dialogfensters von Bedeutung:

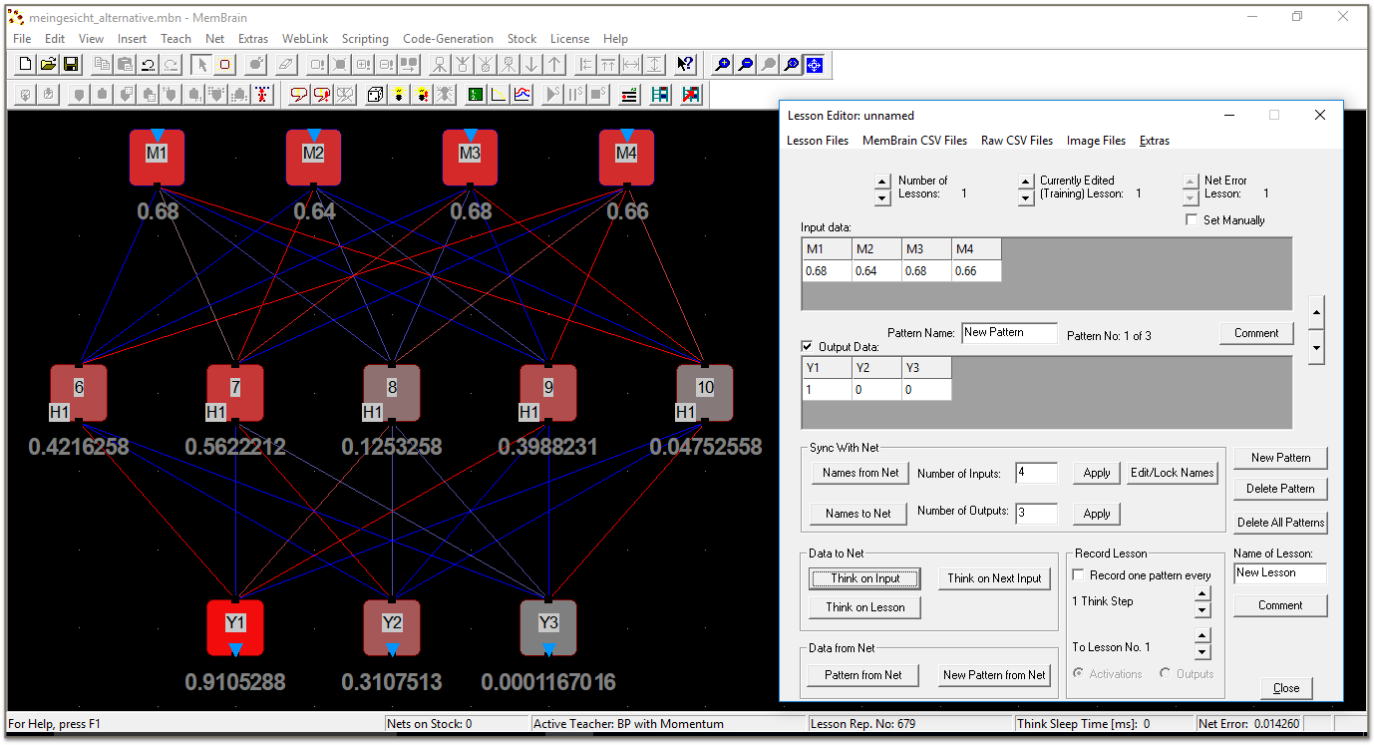
Zunächst wird mit den Pfeil-Buttons rechts das erste Muster ausgewählt, so dass in der Mitte Pattern No. 1 of 3 steht und unter Output Data ebenso eine 0 als Klassenlabel zu sehen ist. Dann bewirkt ein Klick auf den Button Think on Input im Bereich Data to Net, dass das aktuelle Testmuster durch das Netz geschickt wird und das KNN im Haupfenster von MemBrain direkt unter dem Ausgabeneuron anzeigt, welcher Person es das Muster zuordnet. Ein Wert nahe 0 bedeutet also, dass hier auch die Person mit dem Klassenlabel 0 erkannt wurde. Ein Wert von 0.39 würde darauf hindeuten, dass das Netz hinter den aktuellen vier Merkmalen eher die Person mit dem Label 0.5 vermutet als die Person mit dem Label 0. Um welche Person es sich (zum Vergleich) handelt, lässt sich in dem ebenfalls grün markierten Bereich unter Output Data erkennen (die 0 in diesem Fall wird für das KNN schlicht ignoriert und dient nur der eigenen Kontrolle).

Wird nun also die richtige Person erkannt, notieren die SuS dies und wechseln mit den Pfeil-Buttons zum nächsten Muster.

4. Im Anschluss kann dann recht einfach die Erkennungsrate ermittelt werden, die je nach Netztopologie und gewählten Merkmalen tendenziell eher bei 100% liegen sollte.

Tipp: Ein Vergrößern des Datensatzes führt dazu, dass die Erkennungsrate sinken wird, was für den Fortgang der Diskussion über Machine Learning durchaus sehr fruchtbar sein kann.

Alternativen & Gedanken zur hier vorgestellten Gesichtserkennung:



Alternative Modellierung (z.B. für mehrere Klassen)

Eine weitere Möglichkeit, das Klassenlabel zu kodieren, kann darin bestehen, genau die Anzahl an Ausgabeneuronen zu nehmen, die der Anzahl an Personen entspricht. Diese werden in Y1, Y2 und Y3 umbenannt. Die Trainings- und Testdaten müssen dann noch entsprechend angepasst werden, wie in diesem fiktiven Beispiel:

```
M1;M2;M3;M4;Y1;Y2;Y3
0.66;0.64;0.64;0.63;1;0;0
0.69;0.63;0.68;0.66;1;0;0
0.68;0.63;0.67;0.65;1;0;0
0.72;0.66;0.72;0.63;0;1;0
0.73;0.68;0.75;0.64;0;1;0
0.75;0.67;0.75;0.66;0;1;0
0.80;0.69;0.81;0.70;0;0;1
0.82;0.68;0.83;0.70;0;0;1
0.83;0.69;0.84;0.71;0;0;1
```

```
M1;M2;M3;M4;Y1;Y2;Y3
0.68;0.64;0.68;0.66;1;0;0
0.74;0.67;0.75;0.65;0;1;0
0.84;0.69;0.84;0.69;0;0;1
```

So gehört das Muster zur Person 1, wenn der ‚Ausgabevektor' entsprechend (1,0,0) ist, zur Person 2, wenn dieser (0,1,0) ist, und zur Person 3, wenn er (0,0,1) ist.

Weitere Gedanken:

- Ein Netz mit nur drei Neuronen in der verdeckten Schicht mag sogar besser funktionieren.
- Mehr Merkmale führen unter Umständen zu besseren Klassifikationsleistungen.
- Die Verwendung einer anderen Aktivierungsfunktion erhöht möglicherweise die Performance des Netzes (hier kann man schön experimentieren).
- Für solch einfache Anwendungen sind drei Schichten ausreichend, mehr Schichten bringen nicht zwangsläufig eine Verbesserung der Performance.
- Didaktisch interessant mag es sein, einmal eine Verbindung oder ein Neuron zu löschen. In vielen Fällen kann das Netz trotzdem noch erkennen, um welche Person es sich handelt.
- Die Vergrößerung des Testdatensatzes liefert etwas validere Raten.
- Aufgrund der geringen Größe des Datensatzes findet hier ein Overtraining statt, was auch erklärt, warum die Erkennung nahezu immer so gut funktioniert. Echte Anwendungen von Gesichtserkennung werden mit mehr Merkmalen, größeren Netzen und viel mehr Daten erstellt.
- Auch sollte klar werden, dass KNN mit realen Daten so gut umgehen können, weil sie im Gegensatz zum harten Algorithmus fehlertoleranter sind, aber deswegen auch selbst Fehler machen können, daher wäre es für eine sich anschließende Diskussion gut, wenn nicht alle Gruppen eine 100%ige Erkennung hätten. Provozieren lässt sich dass ggf. mit mehr Bildern oder indem ein gemeinsamer Datensatz aus unterschiedlichen Gruppen mit jeweils anderen Gesichtern pro Gruppe erstellt wird.
- Ein schönes Video zum Thema Gesichtserkennung findet man unter autonomousweapons.org, einer Initiative gegen autonome Waffen (s. auch https://www.youtube.com/watch?v=TlO2gcs1YvM)
- Bei all dem sollte niemals der kritische Blick verloren werden. Im Bereich Informatik, Mensch und Gesellschaft können hier großartige Diskussionen entstehen, mit welchen Fragen wir uns in Zukunft auseinandersetzen müssen.