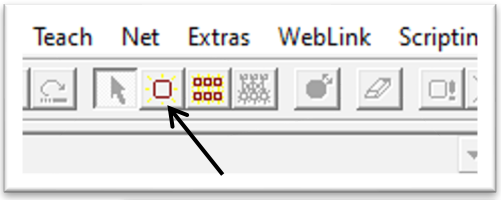
AB 4 - MemBrain - **Lösung**

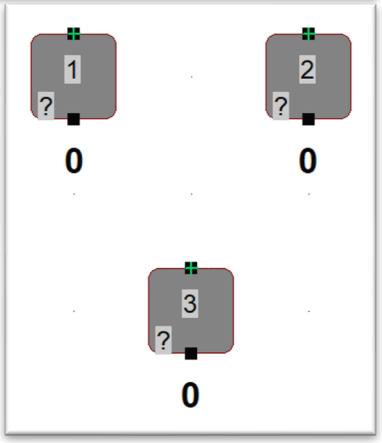
A: Lernalgorithmus einstellen

1. Öffne den Teacher Manager
2. Drücke auf Edit um den voreingestellten Lernalgorithmus zu ändern.
3. Es öffnet sich das Fenster „Edit Teacher“:
   a. Wähle beim Type das zweite von oben → „BP (Full Loopback support)“
   b. Stell bei der Learning Rate 0.4 ein.
   c. Schließe beide Fenster mit Klick auf OK.

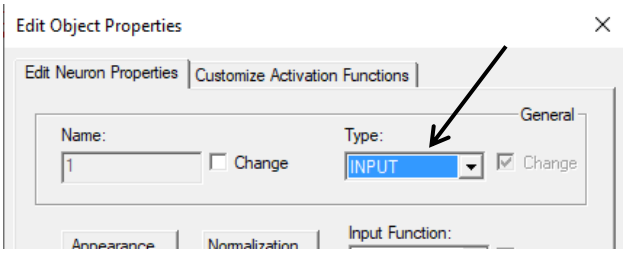
B: Bau eines neuronalen Netzes am Beispiel des booleschen AND

Wichtig: *In MemBrain verlaufen Neuronale Netze von **oben nach unten**, anstatt von links nach rechts!*

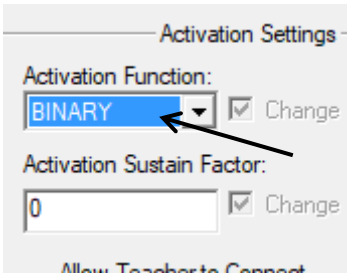




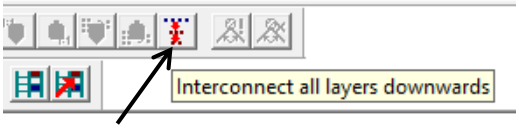
1. Wähle das einzelne Neuron und baue das dargestellte Netz auf. Mit ESC kannst du den Modus wieder verlassen. Einzelne falsche Neuronen kannst du markieren und mit der ENTF-Taste löschen.
2. Markiere nun beide obere (Input-) Neuronen, indem du ein Quadrat um sie ziehst, und mache einen Doppelklick auf eins. Du musst nun einstellen, zu welchem Layer-Typ diese beiden oberen Neuronen gehören. Wähle „INPUT“ und schließe das Fenster anschließend mit OK.



   Nun sollten die beiden oberen Neuronen anders aussehen und als Input-Neuronen erkennbar sein.
3. Du kannst wahlweise die Neuronen auch noch einzeln „doppelklicken“, um die Namen anzupassen (z. B. x1 und x2)
4. Klicke das Output-Neuron doppelt an, um dessen Einstellungen zu öffnen. Wähle dort als Typ „OUTPUT“. Außerdem musst du dort die „Activation Function“ von LOGISTIC auf BINARY ändern. Was das bedeutet, schauen wir uns später an.



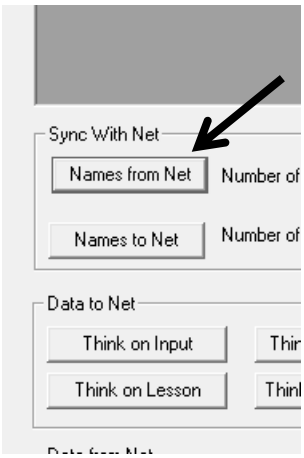
5. Jetzt, nachdem die Neuronen-Layer klar definiert sind, kann man Membrain anweisen, die Neuronen automatisch miteinander zu verknüpfen. Klicke dazu rechts oben auf den rechts gezeigten Knopf.



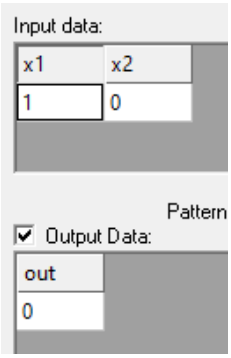
6. Nun sollten beide Inputs mit dem Output verbunden sein (insg. 2 Linien).

Jetzt müssen wir uns noch darum kümmern die Trainingsdaten zu definieren, damit das Netz selbstständig lernen kann die AND-Funktion zu erfüllen.

7. Öffne den „Lesson Editor“ und klicke als erstes auf „Names from Net“, damit der Lesson Editor weiß, welche Neuronen zum Input und welche zum Output gehören. Bestätige mit „Ja“.



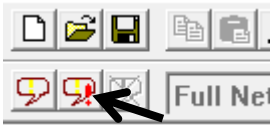
8. Klicke nun rechts am Rand auf „New Pattern“ und trage anschließend die erste Input-/Output-Kombination der AND-Funktion ein (siehe Beispiel-Bild rechts unten).
9. Wiederhole die Schritte aus 8. noch 3 weitere Mal, bis alle 4 Fälle der AND-Funktion vorkommen. Mit den Pfeilen hoch/runter am Rand rechts kannst du deine Eingaben nochmals überprüfen.



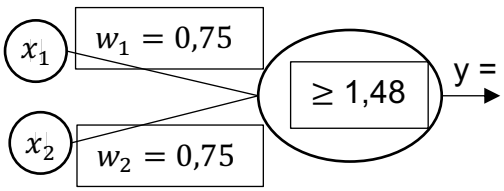
10. Schließe nun den Lesson Editor und öffne den „Net Error Viewer“ (der Knopf direkt rechts vom Lesson Editor).
11. Initialisiere bzw. randomisiere die Gewichte mit zufälligen Werten.

12. Starte nun den automatischen Trainingsvorgang durch das neuronale Netz. Dabei laufen nun genau die gleichen Schritte ab, wie in Aufgabe 5 des letzten Blattes.
13. Du siehst nun im Graphen, wie viele Durchgänge das Netz benötigt hat, um die AND-Funktion perfekt zu beherrschen und wie sich der Verlauf der Fehlerrate entwickelt hat.

14. Du kannst die Schritte 11 bis 13 noch ein paarmal wiederholen um zu sehen, dass der Lernfortschritt teilweise stark vom zufälligen initialen Startwert abhängt.
15. Nun solltest du **überprüfen**, ob das Netz tatsächlich korrekt trainiert wurde. Schließe das kleine Fenster mit dem Graphen und klicke auf den Knopf links in der unteren Zeile.
    Nun kannst du die Input-Neuronen einzeln anwählen und mit der Tastatur 0 oder 1 eingeben. Der berechnete Wert am Output-Neuron wird automatisch angepasst. **<u>Wichtig</u>**: nach der manuellen Überprüfung musst du die „Denkvorgänge“ wieder **stoppen** durch Klick auf den Knopf direkt rechts daneben.



16. Du kannst dir nun anzeigen lassen, welche Werte für die Gewichte und die Aktivierungsschwelle gewählt wurden. Doppelklicke auf die Verbindungen für die Gewichte (→ Weight) bzw. auf das Output-Neuron (→ Activation Treshold).
    Welche Werte hat das Netz automatisch gefunden? Trage sie unten ein (mit etwa 2 Nachkommastellen).



Viele mögliche Lösungen in dieser Aufgabe!

Aufgabe 1:
Trainiere noch nacheinander die OR-Funktion und die NOR-Funktion für die gleichen Neuronen. Dazu musst du im Lesson Editor die bisherigen Patterns löschen und 4 neue Patterns eingeben. Dann kannst du wieder das Netz randomisieren, den Net Error Viewer (den Graphen) öffnen und das Netz automatisch trainieren lassen. Anschließend kannst du wieder manuell überprüfen, ob das Netz wirklich die OR- bzw. die NOR-Funktion korrekt berechnet.

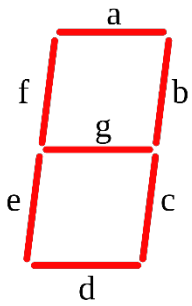
Aufgabe 2a:
Erstelle und trainiere nun ein Netz, das die XNOR-Funktion korrekt darstellt. Funktioniert das mit dem bisherigen simplen Netz aus Aufgabe 1?
Falls nicht: du kannst das bisherige Netz einfach umbauen. Klicke dazu die Verbindungen einzeln an und lösche sie mit der ENTF-Taste. Das Output-Neuron kannst du weiter nach unten ziehen und anschließend Neuronen in einem Hidden-Layer einfügen. Wie viele zusätzliche Neuronen benötigst du in dem Hidden-Layer?
Wichtig:

1. Die neuen Neuronen im Hidden-Layer müssen als Typ „Hidden“ eingestellt sein.
2. Die Aktivierungsfunktion muss im Hidden-Layer (genauso wie beim Output!) manuell auf BINARY gestellt werden.
3. Im „Teacher Manager“ muss die „Learning Rate“ vermutlich stark reduziert werden. Probiere verschiedene Werte der Learning Rate ab 0.01 aus.
4. Falls das Netz nicht zügig (max. ~ 250 Lernschritte) den Fehler auf 0 reduzieren kann, heißt das nicht zwingend, dass die Einstellungen falsch sind. Vielleicht waren nur die zufälligen Startwerte ungünstig gewählt. Probiere daher jede Einstellung einige Male aus (natürlich nachdem du zuvor wieder mit dem Würfel-Knopf die Einstellungen randomisiert hast).

Aufgabe 2b:
Trainiere das obige Netz erneut – verwende nun aber als Aktivierungsfunktion LOGISTIC in den Neuronen des Hidden-Layer sowie im Output-Neuron. Außerdem musst du im „Teacher Manager“ die „Learning Rate“ nun um einiges erhöhen. Wähle dort einen Wert von etwa 3.0.
Was beobachtest du bei der Minimierung des Fehlers? **Zusatzaufgabe**: Erkläre die Beobachtung!
Mögliche Lösung:

Der Fehlerwert strebt schließlich, vergleichbar einer exponentiellen Abnahme, gegen Null. Da die Änderung der Gewichtung abhängig vom Fehler ist, wird die Änderung immer geringer, sodass nie ein Fehler von 0 erreicht werden kann.

Aufgabe 3:



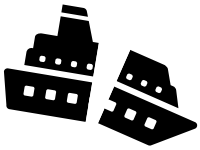
Baue selbst ein neuronales Netz, das eine Binärzahl entgegennimmt und entsprechend dem Zahlenwert eine 7-Segment-Anzeige „aufleuchten“ lassen kann. Überlege dir selbst, wie viele Bits du benötigst, um alle Ziffern mit einer solchen Anzeige darzustellen (1 Bit = 1 Input-Neuron). Jedes Element bekommt ein eigenes Output-Neuron.
Benötigst du einen oder mehrere Hidden-Layer zwischen Ein- und Ausgabeschicht? Als Faustregel für die Anzahl der Neuronen in einem Hidden-Layer gilt: $\left\lceil \frac{1}{2}(Anz.InputNeur. + Anz.OutputNeur.) \right\rceil$
Erstelle selbst alle Lern-Patterns. Die Lernrate sollte auf mindestens 0.2 stehen.

Aufgabe 4:

Begründe, warum man häufig einen Hidden-Layer benötigt. Du kannst am Beispiel von Aufgabe 3 erklären.
Tipp: Was wäre bei Aufgabe 3, wenn es keinen Hidden-Layer gäbe?
Mögliche Lösung:

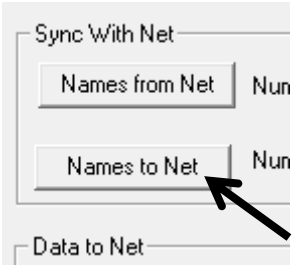
Gäbe es keinen Hidden-Layer, so hätte jedes Bit einen direkten Einfluss darauf, welches Segment der Anzeige jeweils leuchten muss. Damit würden z. B. bei allen ungeraden Zahlen die gleichen Segmente leuchten. Das wäre falsch! Mit einem Hidden-Layer können komplexere Zusammenhänge zwischen den Bits berücksichtigt werden.

Aufgabe 5:



In dieser Aufgabe wirst du ein neuronales Netz mit realen Daten von Passagieren des Titanic-Unglücks trainieren. Du trainierst es mit 891 Datensätzen (siehe Tauschverzeichnis), die jeweils das Geschlecht, das ungefähre Alter und die (Ticket-)Klasse der Person beinhalten sowie den traurigen Fakt, ob jene Person das Unglück überlebte oder nicht. Anschließend kannst du anhand der ersten Parameter recht genau feststellen, ob diejenige Person wohl überlebt hat oder nicht.
Das Alter ist unterteilt in U18, U30, U40, U50 und Ab50. Dabei beinhaltet z. B. U30 alle 18-29-jährigen Passagier/innen.
Die Ticket-Klasse kennt die Werte $0,\overline{3}$ für die 1. Klasse, $0,\overline{6}$ für die 2. Klasse und 1 für die 3. Klasse.
Vorgehensweise:

1. Lade als erstes die CSV-Datei mit den Lern-Datensätzen, indem du im Lesson Editor auf „Raw CSV Files“ → „Import Current Lesson (Raw CSV)“ klickst.
2. Nun siehst du die Struktur, die das neuronale Netz haben muss (Anzahl der Input- und Output-Neuronen). Baue es entsprechend auf – baue zunächst keinen Hidden-Layer ein.
3. Sobald die Anzahl der Input- und Output Neuronen stimmt und die Neuronen auch bereits als Input/Output definiert wurden, gibt es eine Abkürzung. Du kannst nun im Lesson Editor auf „Names to Net“ klicken und musst nicht manuell alle Neuronen genauso benennen, wie es von den Lesson-Patterns vorgegeben ist.
4. Setze im Teacher-Manager die Lernrate auf etwa 0.05!



Sobald das Netz fertig trainiert ist, geht es ans Auswerten:

1. Galt der Verhaltenskodex „Frauen und Kinder zuerst!“?
   a. Teste es manuell, indem du die entsprechenden Input-Neuronen auf 1 setzt und den Output betrachtest.
   Tipp: Damit du die Klasse als 0,333 oder 0,666 eingeben kannst, musst du Doppelklick auf das Neuron machen und den Wert bei „Activation“ eingeben.

   b. Außerdem kannst du diese Aussage überprüfen, indem du dir die Gewichte der Verbindungen anzeigen lässt. Dazu brauchst du nur Doppelklick auf eine Verbindung zu machen.

   Schreibe deine Beobachtungen für a. und b. auf:

Der Verhaltenscodex wurde tatsächlich gut eingehalten. In allen Klassen hatten Frauen bessere Überlebenschancen als Männer, außerdem hatten Kinder bessere Chancen als ältere Leute. Die Verbindungen mit entsprechend positivem Einfluss haben auch ein größeres positives Gewicht und sind rot gefärbt.

2. Verändere nun das Netz, indem du einen Hidden-Layer einfügst und das neue Netz erneut trainierst. Der Fehler des Netzes sollte dadurch etwas geringer werden, man kann aber nun nicht mehr den direkten Einfluss der einzelnen Parameter anhand der Verbindungsgewichte nachvollziehen.
3. Überprüfe für die folgenden sechs Personen, ob diese wohl das Unglück überlebt haben oder nicht.

	Überlebt?	
	Ja	Nein
a. Mr. Anders Johan Andersson, 39 J, 3. Klasse		x
b. Mrs. Mary D Kingcome Hewlett, 55 J, 2. Klasse	x	
c. Miss. Anna McGowan, 15 J, 3. Klasse	x	
d. Mr. Charles Alexander Fortune, 19 J, 1. Klasse		x
e. Mr. Ernest Charles Cann, 21 J, 3. Klasse		x
f. Mrs. Henry Sleeper Harper, 49 J, 1. Klasse	x	
g. Mr. Stephen Curnow Jenkin, 32 J, 2. Klasse		x

4. Versuche eine generelle Aussage bzw. Statistik zu erstellen, welche Menschengruppen eine größere Wahrscheinlichkeit hatten zu überleben:
   Frauen in der 1. Klasse hatten generell eine große Chance – das Alter war weniger entscheidend. In den restlichen Klassen hatten Frauen besonders im Kindesalter große Chancen.
   Männer hatten nur in der 1. Klasse eine größere Überlebenschance – je jünger desto besser. Männer anderer Klassen hatten generell kaum eine Chance.

5. Hättest du die Titanic überlebt?
   Individuelle Antworten

Aufgabe 6 (für die Schnellen):

a) Informiere dich über den Iris-Datensatz (Iris = Schwertlilien) von Ronald Fisher (1936). Beschreibe knapp worum es geht:

Der Iris-Datensatz enthält gemessene Breiten und Längen von Kelchblättern und Kronblättern verschiedener Schwertlilien. Mithilfe von Datenanalyse kann man jede Pflanze einer von 3 Arten zuordnen.

Der Datensatz wird in der Datenanalyse häufig zum Einstieg genutzt und ist gewissermaßen vergleichbar mit einem „Hello-World"-Programm.

b) Lade den Iris-Datensatz aus dem Tauschordner herunter. Erstelle in MemBrain ein neues Netz mit vier Eingabeneuronen und einem Ausgabeneuron. Stelle als Lernalgorithmus „BP with Momentum" mit der Standard-Lernrate von 0.05 ein. Lade den Datensatz in den Lesson Editor von MemBrain und prüfe verschiedene Netzstrukturen auf die niedrigste Fehlerrate (Kein Hidden-Layer, ein Hidden-Layer mit 2, 3, … 5 Neuronen, …).
Die Spezies-Zuordnung ist wie folgt: 0 = Setosa, 0.5 = Versicolor und 1 = Virginica.

Das beste Ergebnis bekommt man scheinbar mit einem Hidden-Layer aus 4 Neuronen → Fehlerrate bis zu < 0.01! Abweichende Lösungen sind aber ebenso möglich, da das Ergebnis auch von den zufälligen Initialwerten abhängt.

c) Zu welcher Spezies gehören die folgenden Pflanzen?

Sepal-Length [cm]	Sepal-Width [cm]	Petal-Length [cm]	Petal-Width [cm]	Spezies
6.1	2.6	5.6	1.4	Virginica
7.3	2.9	6.3	1.8	Virginica
5.7	2.8	4.5	1.3	Versicolor