MatrikelNr:

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Musterlösung + Klausur zur Vorlesung Adaptive Systeme Wintersemester 2013/2014

Datum: 10.02.2014

Vorname:
Name:
Matrikelnummer:
Geburtsdatum:
Studiengang:

Als BSc bearbeiten Sie bitte den Teil der Aufgaben, der mit „AS-1“ gekennzeichnet ist. (max 80 Minuten). Als MSc bearbeiten Sie den“AS-1“ (max. 80 Minuten) und/oder den „AS-2“-Teil (max.100 Minuten, zusammen 180 Minuten). Im AS-1-Teil sind mit 80 Punkten 100% erreicht. Für den AS-2-Teil sind 100 Punkte für 100% nötig.

Die Punktzahl einer Aufgabe entspricht ungefähr der maximalen Bearbeitungsdauer der Aufgabe in Minuten. Die Gesamtpunktzahl der Aufgaben eines Teils übersteigt jeweils 100%, so dass Sie nicht alle Aufgaben lösen müssen.

Durch die Übungspunkte können maximal 10% der Klausurleistung erbracht werden. Als Hilfsmittel ist ein Taschenrechner erlaubt. Bitte benutzen Sie für Notizen die Rückseiten der Aufgabenblätter.

Viel Erfolg!

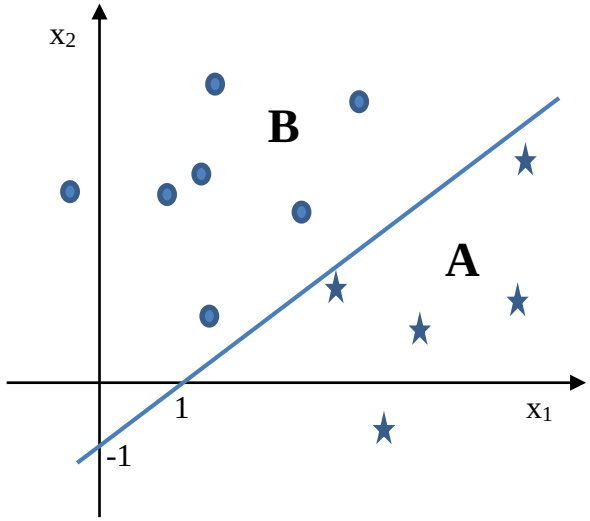
Wird vom Prüfer ausgefüllt:

AS1-1	AS1-2	AS1-3	AS1-4	AS1-5	AS1-6	AS1-7	AS1-8		Σ
/10	/15	/20	/8	/10	/16	/10	/6		/80
AS2-1	AS2-2	AS2-3	AS2-4	AS2-5	AS2-6	AS2-7	AS2-8	AS2-9	
/15	/20	/8	/15	/10	/10	/20	/15	/7	/100

Punkte Klausur: Punkte Übungen: Punkte Gesamt: Note:

AS-1.1 Klassifikation **10 Pkte**

Gegeben sei eine Klassentrennung von Mustern **x** = (x_1,x_2), visualisiert in dem folgenden Diagramm.



Wie lauten die Gewichte, die Aktivität und die Ausgabefunktion für ein binäres Neuron, das die Klassentrennung durchführt? Dazu soll eins für „**x** aus Klasse A“ und null für „**x** aus Klasse B“ ausgegeben werden.

Aus der Zeichnung lässt sich ablesen, dass die Klassentrennung durch eine Gerade mit der Gleichung $x_2 = a\,x_1 + b$ *mit a= 1, b= -1 durchgeführt wird. Also ist die Diskriminanzfunktion* $z(\mathbf{x}^*) = 0 = a\,x_1^* - x_2^* + b = w_1x_1^* + w_2x_2^* + w_0 = (w_0\ w_1\ w_2)(1\ x_1^*\ x_2^*)^T = \mathbf{w}^T\mathbf{x}$ *mit* $\mathbf{w} = (1\ \ 1\ -1)$*. (6 Pkte)*
Ein binäres Neuron mit der Ausgabefunktion S(z) = 1 bei z>0, sonst 0, gibt genau dann 1 aus, wenn $x_1 > x_1^*$ *oder* $x_2 < x_2^*$ *gilt, also* **x** *aus Klasse A ist.(4 Pkte)*

AS-1.2 Perzeptron und Adaline **15 Pkte**

a) Vergleichen Sie die Perzeptron-Lernregel

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \gamma(L - \mathbf{y})\,\mathbf{x}$$

mit der Adaline-Lernregel

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \gamma(L - \mathbf{w}^T\mathbf{x})\,\mathbf{x}\,/|\mathbf{x}|^2$$

Was sind die beiden wesentlichen Unterschiede? (5 Pkte)

Die wesentlichen Unterschiede zwischen beiden Lerngleichungen sind zum einen die Verwendung der Ausgabe $y=S(\mathbf{w}^T\mathbf{x})$ *beim Perzeptron, die eine nicht-lineare Funktion der Aktivität ist im Unterschied zur linearen Ausgabe bei Adaline, (3 Pkte)*
und zum anderen die Normierung der Eingabe $\mathbf{x}\,/|\mathbf{x}|^2$ *bei Widrow-Hoff. Beides zusammen bewirkt eine bessere Konvergenz der Widrow-Hoff-Lernregel. (2 Pkte)*

b) Wie lässt sich eine Konvergenz der Lernregeln erreichen, obwohl die Muster nicht linear separierbar sind? (4 Pkte)

Wenn die Muster nicht linear separierbar sind, findet der Lernalgorithmus keine separierende Trennlinie. Sieht man dagegen ein Abfallen der Lernrate $\gamma(t)$ *mit der Zeit vor, so findet der Algorithmus eine Trennlinie, die den erwarteten quadratischen Fehler*

minimiert, auch wenn er nicht null werden kann. Verändern sich die gefundenen Parameter nicht mehr in der gewählten Rechengenauigkeit, so kann man die Iteration abbrechen.

c) Was versteht man unter „overfitting“ und was kann man dagegen tun? (6 Pkte)

Man bezeichnet als „overfitting“ eine Anpassung der Approximation derart, dass alle beobachteten Werte mit ihren zufälligen Abweichungen von der gesuchten Funktion gut abgedeckt werden, aber die eigentliche Funktion dabei schlecht approximiert wird. Dies macht sich beim Test der Funktionsapproximierung durch unbekannte Beobachtungen bemerkbar. (3 Pkte)
Als Gegenmaßnahme kann man zum einen die Approximation während der Iteration testen und genau dann aufhören, wenn der Testfehler wieder ansteigt (stopped training), oder aber man nutzt für die Approximation eine andere Zielfunktion aus, etwa die Information anstelle der quadratischen Abweichung. In jedem Fall muss man am Ende das gewonnene Ergebnis mit einer von Trainingsprozess unabhängigen Testmenge überprüfen. (3 Pkte)

AS-1.3 PCA und ICA 20 Pkte

Ein schneller Fixpunkt-Algorithmus für PCA besteht aus folgenden Schritten:

1. Zentrieren der gesamten Eingabe $X_1 = \{\mathbf{x}\}$ der Dimension n. Sei i=1.
2. Inititialisieren des Gewichtsvektors $\mathbf{w}_i(t=0)$
3. Bilden der Kovarianzmatrix $C_i = \langle \mathbf{x}\mathbf{x}^T \rangle$ mit $\mathbf{x}$ aus X_i
4. Bilden von $\mathbf{w}_i(t+1) = C_i\, \mathbf{w}_i$ (*Fixpunktiteration*)
5. Orthogonalisieren des Vektors $\mathbf{w}_i$ zu allen anderen vorher gefundenen Vektoren
6. Normieren von $\mathbf{w}_i$ auf den Betrag 1.
7. Führe solange Schritte 4 - 6 aus, bis $\mathbf{w}_i$ zum Eigenvektor $\mathbf{e}_i$ konvergiert ist.
8. Wenn i = n, stop. Ansonsten bilde eine neue Eingabe $X_{i+1} = \{\mathbf{x} \mid \mathbf{x} = \mathbf{x} - (\mathbf{e}_i^T\mathbf{x})\, \mathbf{e}_i\}$ aus der um die Komponenten in $\mathbf{e}_i$-Richtung reduzierten Eingabe.
9. Führe für jede weitere Dimension i = 2..n die Schritte 2 bis 7 aus.

Stellen Sie die für den Fixpunkt-Algorithmus von Hyvärinen zur ICA notwendigen Schritte auf und vergleichen Sie diese mit dem PCA-Fixpunktalgorithmus. Zur Erinnerung: die Fixpunktiteration der ICA war $\mathbf{w}_i(t+1) = \langle (\mathbf{w}_i^T\mathbf{v})^3\mathbf{v} \rangle - 3\mathbf{w}_i$

Welche Schritte sind unterschiedlich, welche gleich, und warum?

Der sequentielle Algorithmus für die ICA sieht folgende Schritte vor: (14 Pkte)

1. *Zentrieren sowie Weissen der gesamten Eingabe {**x**}. Wir erhalten die Mustermenge V0 = {**v**}. (2 Pkte)*
2. *Initialisieren des Gewichtsvektors $\mathbf{w}_i(t=0)$*
3. *Bilden des Erwartungswerts aller $\mathbf{w}^T\mathbf{v}$ von V0 des ersten Gewichtsvektors **w** (Zeile der Matrix W). (2 Pkte)*
4. *Aktualisieren mit der Fixpunktgleichung*
5. *Orthogonalisierung von **w** zu allen bisher erhaltenen Gewichtsvektoren. (2 Pkte)*
6. *Normierung von **w** auf die Länge eins. (2 Pkte)*
7. *Schritte 3-5 sooft bis zur Konvergenz (Fixpunkt). (2 Pkte)*
8. *Wenn alle Dimensionen behandelt wurden, stop. Ansonsten Abziehen der gelernten Raumrichtung **w** von allen Eingabe V0. Dies ergibt V1. (2 Pkte)*
9. *Schritte 2-6 für den zweiten, dritten usw. Gewichtsvektor. (2 Pkte)*

Wir erhalten als Matrix aus den Gewichtsvektoren die Entmischungsmatrix

Vergleich (6 Pkte): Die beiden Algorithmen sind sehr ähnlich, aber auch unterschiedlich. In Schritt 1 muss aber bei der ICA noch die Eingabe noch zusätzlich geweisst werden, um als Gewichtsmatrix eine orthonormale Matrix zu erhalten. (3 Pkte)
*In Schritt 3 wird bei der PCA als Erwartungswert die Kovarianzmatrix aus allen Eingaben **x** gebildet, bei der ICA ist es ein Erwartungswert-Vektor aus allen Eingaben **v**. (3 Pkte)*

AS-1.4 Kohonenkarten 8 Pkte

a) Die Winner-take-all-Regel für Kohonenkarten lautet: Wähle Einheit *k* so, dass
$$|\mathbf{w}_k - \mathbf{x}| = \min_i |\mathbf{w}_i - \mathbf{x}|$$
Wie lautet dagegen die klassische, korrelative winner-take-all-Regel? (3 Pkte)

Die klassische korrelative winner-take-all-Regel lautet: Suche den Klassenprototypen k so, dass
$$\boldsymbol{w}_k\boldsymbol{x} = \max_i \boldsymbol{w}_i\boldsymbol{x}$$

b) Wann stimmen beide überein? (2 Pkte)

Beide Auswahlregeln stimmen überein, wenn die Längen (Beträge) der Prototypen $\boldsymbol{w}_i$ gleich lang sind. In diesem Fall ist $|\boldsymbol{w}_k - \boldsymbol{x}|^2 = \boldsymbol{w}_k^2 - 2\boldsymbol{w}_k\boldsymbol{x} + \boldsymbol{x}^2$ g.d. minimal, wenn $\boldsymbol{w}_k\boldsymbol{x}$ maximal ist.

Geben Sie Beispiele für Nachbarschaftsfunktionen an (3 Pkte)

Bekannte Beispiele sind h(k,i) = 1 für i aus der Nachbarschaft von k, sonst null; sowie $h(k,i) = \exp(-(i-k)^2)$ Gauss'sche Glockenfunktion.

AS-1.5 RBF-Netze 10 Pkte

a) Warum konvergieren RBF-Netze mit schichten-separater Adaption schneller als solche mit Backpropagation-Learning? (5 Pkte)

RBF-Netze mit separater Adaption konvergieren schneller, da die Adaption jeder Schicht auf dem Endzustand der vorhergehenden Schicht aufbaut. Jede Schicht konvergiert maximal schnell, da die Verteilung der Eingabevariablen sich nicht ändert. Werden dagegen die Schichten zur gleichen Zeit adaptiert, so ändert die Adaption einer Schicht die gesamte Eingabe der darauf folgenden Schicht und behindert die Konvergenz zum endgültigen Zustand.

b) Warum lernen RBF-Netze schlechter als Multilayer-Perzeptrons mit sigmoidalen Ausgabefunktionen, wenn sie extrapolieren müssen? (5 Pkte)

Extrapolation bedeutet Eingaben, die nicht im „normalen" Eingabebereich lagen. RBF-Funktionen reagieren nur auf Eingaben in ihrem Einzugsbereich. Müssen sie eine Aktivität zu einer Eingabe erzeugen, auf die sie vorher nie trainiert wurden, so sind die Approximationsleistungen sehr schlecht. Im Gegenteil dazu reagieren sigmoidale Ausgabefunktionen immer auf den gesamten möglichen Eingabereich und werden damit auch implizit auf ungewöhnliche Eingaben trainiert und haben so bessere Leistungen bei Extrapolation.

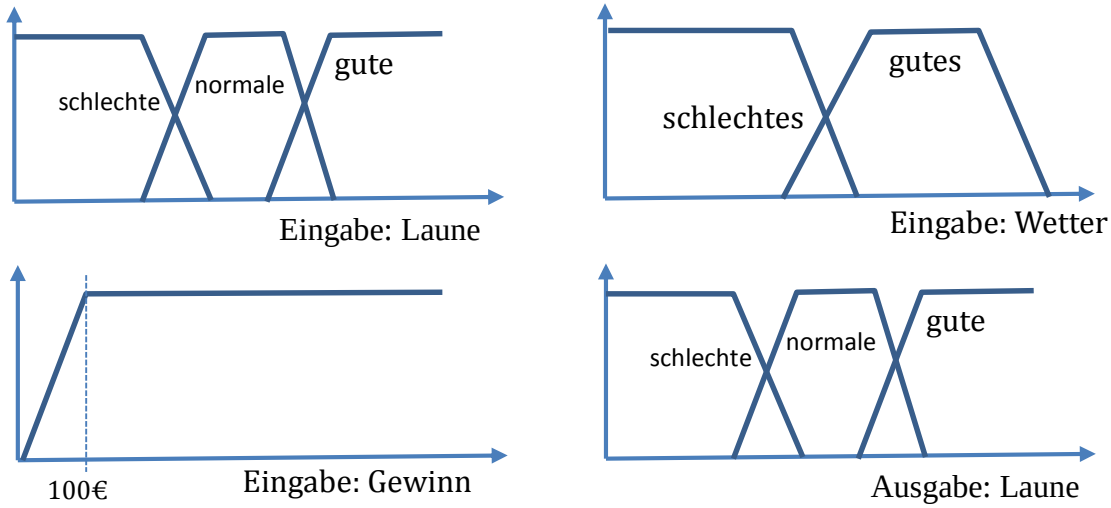
MatrikelNr:

AS-1.6 Fuzzy-Regelung 16 Punkte

Angenommen, Sie haben drei Gemütszustände: schlechte Laune, normal, gute Laune sowie zwei Wahrnehmungen: im_Lotto_gewonnen und das Wetter mit den Zuständen (schlecht, gut).

a) Entwerfen Sie dazu plausible Zugehörigkeitsfunktionen (8 Pkte)
b) sowie eine plausible Fuzzy-Regelung für Ihren Gemütszustand. Sehen Sie dabei alle möglichen Prinzipien vor. (8 Pkte)

a) Wir haben die Fuzzy-Variablen „Gemütszustände" mit drei Zuständen {schlechte_Laune, normale_Laune, gute_Laune}, die zwei Wetterzustände {schlechtes_Wetter, gutes_Wetter} sowie eine Variable {im_Lotto_gewonnen}. Die plausiblen Zugehörigkeitsfunktionen sind (je 2 Pkte)



Die Laune wird nur mit drei Zuständen modelliert; das Wetter kann aber auch sehr gutes und Himmlisches Wetter enthalten, was aber hier nicht gefragt ist. Der Lottogewinn wird erst ab 100 € als solcher bezeichnet, vorher wird er nicht als Gewinn angesehen, sondern z.B. als Gebührenrückzahlung, was aber hier nicht modelliert wird. Da dies eine Regelung der Laune ist und von der Laune auf die Laune eingewirkt wird, ist die Ausgabevariable genauso modelliert wie die korrespondierende Eingabe.

*b) Die Fuzzy-Regelung soll dies durch Prinzipien regeln. Also müssen wir 3*2*1 = 6 Eingaben sowie 3 Ausgaben verarbeiten. (2 Pkte)*
Als 6 Prinzipien können also aufstellen: (je 1 Pkte)
WENN schlechte_Laune UND schlechtes_Wetter UND im_Lotto_gewonnen DANN schlechte_Laune
WENN schlechte_Laune UND gutes_Wetter UND im_Lotto_gewonnen DANN normale_Laune
WENN normale_Laune UND schlechtes_Wetter UND im_Lotto_gewonnen DANN normale_Laune
WENN normale_Laune UND gutes_Wetter UND im_Lotto_gewonnen DANN gute_Laune
WENN gute_Laune UND schlechtes_Wetter UND im_Lotto_gewonnen DANN gute_Laune
WENN gute_Laune UND gutes_Wetter UND im_Lotto_gewonnen DANN gute_Laune

AS-1.7 Evolutionäre Algorithmen 10 Punkte

Geben Sie die grundsätzlichen algorithmischen Iterationsschritte an, um mit Hilfe eines evolutionären Algorithmus das Optimum einer Zielfunktion zu finden.

Für einen evolutionären Algorithmus benötige ich eine Menge potentieller Lösungen G = {g_1,..,g_n}. Jede Lösung g_i wird durch ein Tupel von Parametern charakterisiert und hat eine Güte R(g_i). *(2 Pkte)*
Die Iteration erfolgt in den Schritten

1. *Bilde die Güte aller Lösungen* *(2 Pkte)*
2. *Suche die besten s Lösungen heraus, etwa s=n/2.* *(2 Pkte)*
3. *Ist eine ausreichende Lösung darunter, stop.* *(1 Pkt)*
4. *Ansonsten bilde n-s neue Lösungen durch Modifikation der besten Lösungen, etwa durch Mutation der Parameter, also zufällige Veränderung der Parameterwerte. Bei reellen Zahlen bietet sich eine Veränderung gemäß einer Gauß'schen Glockenkurve an, also eine normal verteilte Abweichung.* *(3 Pkte)*
5. *Wiederhole Schritte 1 – 4.*

Man beachte: eine einzige iterierte Lösung G = {g} ist nur ein Spezialfall des allgemeinen evolutionären Algorithmus und wird deshalb mit 8Punkten nicht voll bewertet..

AS-1.8 Online vs. Offline-Learning 6 Pkte

a) Was ist der Unterschied zwischen offline-learning und online-learning? (3 Pkte)

Beim Offline-Learning sind alle Muster zum Training vorhanden. Beim Online-Algorithmus treten die Trainingsmuster nacheinander auf und werden nicht gespeichert. (2 Pkte)
Zum Lernen werden beim Online-Learning nach jedem Trainingsmuster sofort die Gewichte verbessert. Im Gegensatz dazu werden beim Offline-Learning die benötigten Gewichtsverbesserungen nur erfasst und aufsummiert. Nach dem Durchlauf aller Trainingsdaten wird dann in einem Schritt die akkumulierte Gewichtsverbesserung angewendet, so dass hier die Verbesserung den Erwartungswert darstellt. (1 Pkt)

b) Welche Methode konvergiert meist schneller und warum? (3Pkte)

Meist konvergiert Online-Learning schneller, da bei jedem Schritt das System verbessert wird und der Fehler gegenüber dem verbesserten System erfasst wird und nicht gegenüber dem Originalsystem. (2 Pkte)
Allerdings kann es bei stark schwankender Statistik der Eingaben zu Fehladaptionen beim Online-Learning kommen, so dass in diesem Fall das Offline-Learning mit der Erwartungswertbildung besser abschneiden kann. (1 Pkt)

Ende des Teils AS-1

Beginn Teil AS-2

AS-2.1 Formale Neuronen und Gatter 15 Pkte

Geben Sie die Werte für die Gewichte für ein zweischichtiges neuronales Netz an, welches eine boolsche Funktion F implementiert. Diese sei für drei Eingaben x_1, x_2, x_3 folgendermaßen definiert: $F(x_1, x_2, x_3) = \overline{x_1}x_2x_3 + x_1\overline{x_2}x_3 + x_1x_2\overline{x_3}$

Im *hidden Layer* sollen drei binäre Neuronen mit $S(z) = \begin{cases} 1 & z > 0{,}5 \\ 0 & z \leq 0{,}5 \end{cases}$ zum Einsatz kommen.

Die Ausgabe wird von einem linearen Neuron erzeugt.

Die Ausgabe ist $F = w_0 + w_1S_1 + w_3S_2 + w_3S_3$. Wählen wir $w_0=0$, $w_1=w_2=w_3=1$ so ist F = 1, wenn einer der drei Ausgänge S_i der ersten Schicht eins ist. (5 Pkte)
Nun müssen wir nur noch dafür sorgen, dass nur bei Vorliegen von einer der drei Zustände (0 1 1), (1 0 1) und (1 1 0) eine Ausgabe $S_i=1$ erfolgt. Sei $S_i(z)$ wie oben gegeben, so ist dies bei $z = w_0 + w_1x_1 + w_2x_2 + w_3x_3$ mit $w_0 = 0$, $w_i=-1$ und alle anderen Gewichte w=0,5 der Fall. Warum? Ist $x_i=1$, so ist selbst bei Vorliegen aller anderen x=1 die Summe $z \leq 0$ und damit $S_i = 0$. Erst wenn $x_i=0$ ist und alle anderen Eingaben eins sind, ergibt z den Wert 1 >0,5. Damit wird genau dann eine Ausgabe des hidden layer *eins, wenn eine der drei genannten Eingabezustände vorliegt. (10 Pkte)*

AS-2.2 Klassifizierung und stochastische Mustererkennung 20 Pkte

a) Gegeben seien die bedingten Wahrscheinlichkeiten von Mustern und Klassen. Wie lautet die optimale Klassifikationsregel? (4P)

Die beste Klassifikation wird durch die Bayes-Klassifikation erreicht: Wähle diejenige Klasse, die die größte bedingte Wahrscheinlichkeit P(Klasse | Muster x liegt vor) hat. (4 Pkte)

b) Welche Maße kennen Sie, um die Güte eines Klassifikationssystems mit Hilfe einer einzigen reellen Zahl zu beschreiben? (6P)

Das klassische Maß ist die Fläche unter der ROC-Kurve (AUC). Sie wird im besten Fall zu eins, im schlechtesten zu 0,5. (2 Pkte)

Aber auch die equal error rate EER wird verwendet, die angibt, welchen Wert Sensitivität und Spezifität haben, wenn sie gleich sind. (2 Pkte)

Ein weiteres populäres Maß ist die mittlere korrekte Diagnose, die sich aus der Summe der Wahrscheinlichkeiten [P(Klasse_liegt_vor diagnostiziert / Klasse_liegt_vor) + P(Klasse_liegt_nicht_vor diagnostiziert / Klasse_liegt_nicht_vor)] /2 ergibt. (2 Pkte)

c) Angenommen, ein Diagnosesystem wird durch einen Parameter p gesteuert. Welche Schritte müssen Sie durchführen, um die ROC des Diagnosesystems zu bestimmen? (10P)

Für die ROC-Kurve müssen eine Menge von Einzelpunkten bestimmt werden. Dazu wird der Gesamtbereich des Parameters so aufgeteilt in einzelne Intervalle, dass jedes Intervall ein anderes Tupel von Wahrscheinlichkeiten (Sensitivität, Spezifität) der Diagnose ergibt. Beispielsweise kann der Parameter ein Schwellwert sein, dessen Überschreiten die Existenz einer Klasse bedeutet. (4 Pkte)

Für jeden Parameterwert muss die gesamte Testmenge verwendet werden, um jeweils

einen Punkt (ein Tupel) zu gewinnen. (2 Pkte)

Die Gesamtheit aller Einzelpunkte (aller Tupel) werden mittels einer Kurve approximiert. Dies ist die ROC-Kurve. (4 Pkte)

AS-2.3 Approximationseigenschaften Neuronaler Netze 8 Pkte

Angenommen, Sie haben ein neuronales Netz.

a) Wie gut kann das Netz eine vorgegebene Funktion approximieren? (3 Pkte)

Ein zweischichtiges Netz kann jede beliebige Funktion beliebig dicht approximieren.

b) Was sind die Vorbedingungen dafür? (5 Pkte)

Dies ist aber nur bei genügend großer Anzahl von Neuronen der Fall. (2 Pkte)

Weiterhin muss die erste Schicht nicht-lineare Einheiten haben, etwa sigmoidale oder Glocken-Ausgabefunktionen. Die zweite Schicht kann linear sein, muss aber nicht. (3 Pkte)

AS-2.4 Training und Testen 15 Pkte

Angenommen, Sie haben von 3 Klassen je 10 Datenquellen (Individuen) und von jeder Quelle 3 Muster für Training, Testen und Validieren.

a) Wie müssen Sie Trainings- und Testmengen bilden und was müssen Sie dabei beachten? (5 Pkte)

Die Mustermenge wird in 2 (Training und Testen) oder 3 (Training, Testen, Validieren) Mengen unterteilt, je nach Aufgabenstellung. (2 Pkte) Dabei ist zu beachten, dass alle Samples einer Quelle in derselben Menge liegen, um einen Transfer der Information über die Quelle zwischen den Mengen, etwa Training und Testen, zu verhindern. (3 Pkte)

b) Wie funktioniert die Kreuzvalidierung? (5 Pkte)

Haben wir zu wenig Muster in einer Menge, so können wir die Anzahl dadurch erhöhen, dass wir fast alle Muster zum Training verwenden und nur wenige (minimal 1) zum Testen. (2 Pkte) Allerdings müssen wir dann das Training wiederholen, wobei jeweils die Testmenge anders zusammengestellt wird. (1 Pkt) Am Schluss werden alle Testergebnisse über die Anzahl der Testläufe gemittelt.(2 Pkte)

c) Wie funktioniert „stopped training“? (5 Pkte)

Bemerken wir einen starken Unterschied zwischen den Trainingsergebnissen und den Testergebnissen, so liegt ein overfitting *vor. In diesem Fall können wir nach jeder Iteration einen Test mit allen Mustern einer Validierungsmenge durchführen und das Training genau dann abbrechen, wenn der Validierungsfehler nicht mehr ab-, sondern zunimmt. (3 Pkte)*

Allerdings muss der Erfolg noch durch eine weitere, vom Training unabhängige Testmenge überprüft werden. (2 Pkte)

AS-2.5 TLMSE 10 Pkte

a) Was ist der Total Least Mean Squared Error? (4 Pkte)

Der TLMSE misst die quadratische Abweichung als kürzesten Abstand der Datenpunkte zu der approximierenden Hyperebene.

b) Warum ist er ein besseres Maß für die Güte einer linearen Approximation als der „normale“ quadratische Fehler MSE? (3 Pkte)

Im Unterschied zum MSE ist er unabhängig von der Lage des Koordinatensystems, ist also invariant bezüglich der Rotation der Basisvektoren.

c) Wie wird die approximierende Hyperebene mit kleinstem TLMSE berechnet? (3 Pkte)

Zur Berechnung wird der Eigenvektor der Daten gebildet mit dem kleinsten Eigenwert. Dieser Vektor steht senkrecht auf der approximierenden Hyperebene und dient der Beschreibung der Hyperebene mittels der Hesseschen Normalform.

AS-2.6 Ljapunov-Funktion 10 Pkte

Was ist eine Ljapunov-Funktion und wozu benötigt man sie? (4 Pkte)

Eine Ljapunov-Funktion R(t) nimmt mit der Zeit monoton ab ($R'(t) \leq 0$) und hat einen minimalen Wert. (2 Pkte)
Hat eine gegebene Funktion diese Charakteristika, beispielsweise die Zielfunktion bei einer Iteration, so muss die Iteration konvergieren. Mit dem Nachweis der Ljapunov-Eigenschaft ist also auch die Konvergenz bewiesen. (2 Pkte)

a) Warum erfüllt der Gradientenabstieg die Ljapunov-Bedingung? (6 Pkte)

Der Gradientenabstieg einer Zielfunktion $R(t) \geq 0$ ist mit $\Delta w = -g \partial R(w(t))/\partial w$ pro Zeiteinheit $\Delta w/\Delta t \approx w'(t) = -gR'(w)$. Dies bedeutet für die Ableitung der Zielfunktion $R'(t) = R'(w(t)) = R'(w)\, w'(t) = -R'(w) \cdot gR'(w) = -g\, R'(w)^2$. Da $g>0$ und $R'^2 \geq 0$ ist beim Gradientenalgorithmus immer $R'(t) \leq 0$. Da R(t) auch ein absolutes Minimum hat, sind die Ljapunov-Eigenschaften immer gegeben und die Iteration immer konvergent.

AS-2.7 Lagrange-Optimierung 20 Pkte

Angenommen, Sie haben Signale, die nur positiv sein können wie etwa die Spikefrequenz von Neuronen, und modellieren sie mit einer stochastischen Variablen x aus [0, +∞]. Welche Wahrscheinlichkeitsverteilung p*(x) trägt die meiste Information unter allen Funktionen p(x), die endlichen Mittelwert *m* haben? Leiten Sie sich die Form von p her mit Hilfe der Lagrang'schen Methode und beachten Sie dabei die beiden Nebenbedingungen, die Wahrscheinlichkeitsdichte p(x) ist normiert auf eins $\langle 1 \rangle = 1$, und der Mittelwert ist endlich $\langle x \rangle = m$.

Die mittlere Information der Verteilung p(x) ist $H(p) = \langle -\log p \rangle$. Ihr Maximum unter den Nebenbedingungen nimmt sie an, wenn mit den Nebenbedingungen $(\langle 1 \rangle - 1) = 0$ und $(\langle x \rangle - m) = 0$ die Ableitung der Lagrangefunktion

$L(p, \mu_1, \mu_2) = H(p) + \mu_1(\langle 1 \rangle - 1) + \lambda_2(\langle x \rangle - m)$ *(6 Pkte)*

ein Extremum annimmt. Dies ist gegeben bei $\partial L/\partial p = 0$, also

$\frac{\partial}{\partial p} L = \frac{\partial}{\partial p} \int_0^\infty -p \log p + \lambda_1 p + \lambda_2 x p \, dx = 0$ *(4 Pkte)*

Dies ist nur dann immer null bei $1>p>0$ für alle positive x, wenn alle Terme im Integral null ergeben. Also ist

$-\frac{\partial}{\partial p} p \log p + \frac{\partial}{\partial p} \lambda_1 p + \frac{\partial}{\partial p} \lambda_2 x p = 0$ *oder* *(6 Pkte)*

$-\log p - 1 + \lambda_1 + \lambda_2 x = 0$ *oder*

$-\log p - 1 + \lambda_1 + \lambda_2 x = 0$ *oder*

$\log p = \lambda_1 - 1 + \lambda_2 x$ *bzw.* $p(x) = \alpha \exp(\lambda_2 x)$ *(4 Pkte)*

Da das Integral von p(x) endlich ist, muss $\lambda_2 \leq 0$ gelten. Die gesuchte Verteilung maximaler Information bei x aus [0, +∞] ist also eine einfache fallende Exponentialverteilung, keine Gaußverteilung wie für x aus]-∞,+∞[oder eine uniforme Verteilung wie für x aus [0,1].

AS-2.8 Fixpunktgleichungen 15 Pkte

Angenommen, Sie haben eine Iteration der Form

$$w(t+1) = w(t) - \gamma(w(t) - a)w(t) - 9 \quad \text{mit } \gamma = 4,\ a = 5$$

Zu welchen Werten w* konvergiert diese Iteration? Bestimmen Sie dazu die stabilen und labilen Fixpunkte.

Zur Erinnerung: Die Lösung von $x^2 + px + q = 0$ ist $x_{1,2} = -p/2 \pm \sqrt{(p/4)^2 - q}$

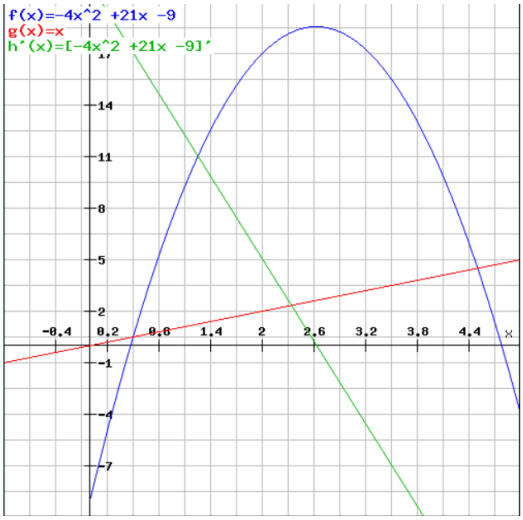
Die Fixpunkte ergeben sich durch die Fixpunktbedingung

$w^* = f(w^*) = w^* - 4(w^* - 5)w^* - 9$

oder $-4w^{*2} + 20w^* - 9 = 0$

$w^{*2} - 5w^* + 9/4 = 0$

Dies ist eine Parabel, die einen Scheitelpunkt oben hat.



Die Lösung der quadratischen Gleichung ist $w^*_{1,2} = 2{,}5 \pm \sqrt{25/4 - 9/4} = 2{,}5 \pm 2$

Also sind Fixpunkte bei w = 0,5 und 4,5. (10 Pkte)*

Da die Ableitung f'(w) = 1 -8w +21 ist (1 Pkt), ergibt sich an den Fixpunkten bei

w=0,5 zu f'(w*) = 1 - 4 + 20 = +17. Da |+17|>1 ein labiler Fixpunkt (2 Pkte), und bei*

w=4,5 zu f'(w*) = 1-36 + 20 = -15. Da |-15|>1 auch ein labiler Fixpunkt. (2 Pkte)*

Die Iteration konvergiert nicht zu finiten Werten: Innerhalb des Intervalls [0,5 4,5] wird die Iteration bei w<2,6 nachts rechts und bei w>2,6 nach links streben und so nie konvergieren.

AS-2.9 RBF 7 Pkte

Was sind die wesentlichen Vorteile und Nachteile von RBF-Netzen gegenüber Multi-Layer-Perzeptrons?

Vorteile: schichtweises, einfaches Training (1 Pkt), schnellere Konvergenz (1 Pkt), inkrementierbares Netz (2 Pkte), weniger Neuronen pro Klassenentscheidung bei geclusterter Eingabe. (1 Pkt)

Nachteile: schlechte Extrapolationsmöglichkeiten (2P)