Blatt 02 - Assoziativspeicher

Abgabe: Bis Mittwoch, 5.November 2014, 10:00 Uhr

An: Tobias Rothenberger, <rothenb@informatik.uni-frankfurt.de>

Aufgaben:	Aufgabe 1			Aufgabe 2		
	1a	1b	1c	2a	2b	2c

Aufgabe 2.1

In dieser Aufgabe werden wir uns als erstes den Hebb'schen Assoziativspeicher etwas näher ansehen. Wir werden dabei die folgenden Vektoren benötigen:

$$a_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad a_2 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \quad a_3 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \quad a_4 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \quad a_5 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad a_6 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \quad a_7 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \quad a_8 =$$

$$b_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \quad b_2 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \quad b_3 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \quad b_4 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

a. Implementieren Sie einen Hebb'schen Assoziativspeicher, der die Operationen "Einfügen", "Auslesen" und "Löschen" von Musterpaaren unterstützt.

   Speichern Sie dann die Paare (a_1, b_1), (a_2, b_2), (a_3, b_3) und (a_4, b_4) in Ihr Assoziativspeicher ab und weisen Sie durch Berechnen der Ausgabe

Eingabe von a_1, a_2, a_3, a_4 nach, dass alle Paare korrekt gespeichert wurden.

Wie lautet die zugehörige Speichermatrix des Hebb'schen Assoziativspeichers (die Matrix der Verbindungsgewichte der Neuronen)? Welche Ausgabe erhalten Sie, wenn Sie statt a_1, a_2, a_3, a_4 einige Eingabevektoren $a'_1, a'_2, a'_3, a'_4 \in \{\pm 1\}^7$ wählen, die sich an einer einzelnen Stelle von a_1, a_2, a_3 oder a_4 unterscheiden?

nach oben

b. Wie sich leicht nachprüfen läßt, sind die Vektoren a_1, a_2, a_3, a_4 weitestgehend orthogonal zueinander, so dass der Störterm beim Auslesen von b_1, b_2, b_3, b_4 keine Rolle mehr spielt (siehe Einleitung). Damit ist garantiert, dass sich b_1, b_2, b_3, b_4 fehlerfrei wieder auslesen lassen. Man kann leicht nachrechnen, dass auch b_1, b_2, b_3, b_4 orthogonal zueinander sind, so dass sich auch eine Speichermatrix für die Paare (b_1, a_1), $(b_2,$ (b_3, a_3) und (b_4, a_4) angeben läßt. Berechnen Sie diese Matrix und vergleichen Sie sie mit der aus Aufgabenteil a). Was stellen Sie fest und warum ist dies so?

Verwenden Sie dieses Ergebnis, um Ihren Assoziativspeicher aus Aufgabenteil a) so zu erweitern, dass er bei Eingabe von unvollständigen Paaren $(a_i, \cdot)$ oder $(\cdot, b_j)$ $(1 \leq i, j \leq 4)$ das fehlende Tupelelement rekonstruieren kann.

nach oben

c. Im Hebb'schen Assoziativspeicher lassen sich auch Tupel beliebiger

Länge speichern. Um beispielsweise das Tupel (a_1, a_2, a_3, a_4) abzuspeichern, genügt es, die Paare (a_1, a_2), (a_2, a_3), (a_3, a_4) und $(a_4, a$ den Assoziativspeicher einzuspeisen. Bei Eingabe von a_1 berechnet das Netz dann als Ausgabe a_2. a_2 kann dann wiederum als Eingabe aufgefaßt und die Ausgabe a_3 dazu berechnet werden, usw. Insbesondere läßt sich so auch ein längeres unvollständiges Tupel rekonstruieren (solange die Rekonstruktion eindeutig ist).

Erweitern Sie Ihren Assoziativspeicher nun so, dass er bei Eingabe beliebig langer unvollständiger Tupel die fehlenden Tupelelemente so weit wie möglich rekonstruiert.

Betrachten Sie dann die folgende Speichermatrix W und verwenden Sie Ihr Programm, um zu dieser Matrix herauszufinden, wie die fehlenden Tupelelemente der beiden unvollständigen Tupel $(\cdot, a_3, \cdot)$ und $(\cdot, a_7, \cdot, \cdot,$ aussehen.

$$W = \begin{bmatrix} -3 & -3 & 5 & -3 & 1 & 1 & 1 \\ 3 & -1 & 3 & -1 & -5 & -1 & 3 \\ -5 & -1 & -5 & -1 & -5 & -1 & 3 \\ -3 & 5 & 5 & 5 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & -7 & 1 \\ -1 & -5 & -1 & 3 & 3 & -1 & 3 \\ -1 & -5 & -1 & 3 & -5 & -1 & -5 \end{bmatrix}$$

nach oben

Aufgabe 2.2 - Assoziativspeicher zur Musterrekonstruktion

a. Die Datei `digits.dt` enthält 10 Schwarz-Weiß-Graphiken der Ziffern 0 bis 9 mit einer Größe von 8x10 Pixel. Lesen Sie diese Graphiken ein und trainieren Sie ein AdaLinE mit dem Widrow-Hoff-Algorithmus im Online-Modus darauf, bei Eingabe einer beliebigen Ziffer, diese wieder als Ausgabe zu produzieren. Trainieren Sie das Netz dabei auch mit leicht verrauschten Eingaben (aber der zugehörigen korrekten, nicht verrauschten Ausgabe) und verwenden Sie als Ausgabefunktion der Neuronen die Funktion $S(z) = \begin{cases} +1, & z \geq 0 \\ -1, & z < 0 \end{cases}$.

   Das Netz wird auf diese Weise zu einem sogenannten *Autoassoziativspeicher* (d.h einem Speicher, bei dem die Muster jeweils zu sich selbst assoziiert sind), in dem die 10 Ziffern eingespeichert sind. Nach erfolgreichem Training wird das Netz dann einige interessante Eigenschaften besitzen, wie wir gleich sehen werden. Durch die leicht verrauschten Eingaben während des Trainings wird das Netz versuchen, die Gewichtsmatrix derart anzupassen, dass es gewissermaßen als autoassoziativer "Musterwiederhersteller" verwendet werden kann.

   Nachdem Sie das Netz ausreichend trainiert haben (d.h. der Fehler des Netzes sich nicht mehr wesentlich senken lässt - ein gutes Kriterium hierfür ist, dass sich die Länge des Gradientenvektors der Fehlerfunktion dem Wert Null nähert, siehe Einleitung), erzeugen Sie zu jeder der 10 Ziffern eine verrauschte Version, bei der etwa 10 Prozent der Pixel zufällig invertiert wurden. Berechnen Sie dann die Ausgabe des Netzes bei Eingabe dieser verrauschten Ziffern und stellen Sie dann jeweils die verrauschte, die korrekte und die von ihrem Netz rekonstruierte Ziffer graphisch dar. (Es empfielt sich hier eine Graphik anzufertigen, in der die 10 verrauschten Ziffern nebeneinander in einer Reihe und die

ursprünglichen und rekonstruierten Ziffern jeweils darunter angeordnet sind.)

nach oben

b. Das Trainieren des Netzes mit fehlerhaften Ziffern hat auch den interessanten Nebeneffekt, dass die Gewichtsmatrix selbst robuster gegenüber Fehlern wird. Berechnen Sie nun für $p = 0, 1, 2, ..., 50$ die Ausgabe des Netzes bei Eingabe aller 10 Ziffern, wenn etwa p Prozent der Einträge der Gewichtsmatrix gelöscht (auf Null gesetzt) werden. Bestimmen Sie dann die *erwartete* Anzahl aller fehlerhaften Pixel, die Sie erhalten (d.h. wenn Sie bei Eingabe der i-ten Ziffer a_i Pixelfehler erhalten, dann ist der Erwartungswert $\left\langle \sum_{i=1}^{10} a_i \right\rangle$ zu berechnen).

Stellen Sie das Ergebnis anschließend in einem Plot dar. Wieviel Prozent der Gewichtsmatrix kann in etwa auf diese Weise gelöscht werden, so dass im Erwartungswert unter allen Ziffern höchstens 1 Pixel fehlerhaft ausgelesen wird?

nach oben

c. Möglicherweise konnten Sie in Aufgabenteil 2a) feststellen, dass gelegentlich noch einzelne Pixelfehler beim Auslesen der Ziffern vorhanden sind. Wenn Sie statt 10 Prozent nun 20 Prozent der Pixel einer Ziffer invertieren, werden Sie feststellen, dass diese kaum noch vollständig korrekt ausgelesen werden können. Haben Sie eine Idee, wie dieses Ergebnis auf einfache Weise verbessert werden kann? (zeigen Sie anhand einiger Beispiele, dass Ihr Verfahren auch wirklich funktioniert)

nach oben