Vorgehen

Hinweise zu: Aufgabe 6.1 Aufgabe 6.2 Aufgabe 6.3 Aufgabe 6.4

Hinweise zu Aufgabe 6.1

Testen Sie Ihr Netzwerk schichtweise. Verwenden Sie dazu für die Aktivität als erste Schicht das Adaline-Netz, als zweite Schicht nur ein lineares Netz.
Für den Test der Aktivität können Sie die Ausgabefunktion per Hand auf die Identität setzen, die Gewichte per Hand vorbestimmen (z.B. bei einer 2x2 Matrix auf die Werte (0.1, 0.2, 0.3, 0.4) setzen) und ein einfaches Beispiel einer linearen Transformation eingeben, etwa x = (1.0, 2.0), und die Ausgabe (0.5, 1.1) per Hand prüfen. Das Analoge gilt auch für die zweite Schicht.

Das Lernen testen Sie am Besten wieder mit einer einfachen Vorgabe einer Eingabe, eines Ziels und einfachen, vorgegeben Gewichtswerten. Rechnen Sie sich die neuen Gewichtswerte vorher aus und überprüfen Sie diese danach.

Für die laufende Überwachung des Lernens ist es hilfreich, die Werte für die Aktivitätskorrektur (das Delta) sowie für die Gewichtskorrektur (den aktuellen Wert des Gradienten) auszudrucken bzw. als Graph zu plotten.

Es gibt hier mehrere stabile Endzustände, die das Netz erreichen kann und die alle recht ähnliche Wirkungen haben. Sie sollten aber das Training wiederholen, wenn Sie das Gefühl haben, daß es noch einen "besseren" Zustand geben kann.

nach oben

Hinweise zu Aufgabe 6.2

Verwenden Sie 16 Neuronen im *Hidden layer*, 50000 Trainingsepochen und eine Lernrate von 0:01. Die Neuronen im *Hidden Layer*
sollen dabei die *tanh*-Funktion als Ausgabefunktion besitzen, die 9 Neuronen der Ausgabeschicht eine lineare Ausgabefunktion der Form S(z) = z. Die Gewichte des Netzes können Sie im Backpropagation-Algorithmus mit zufälligen Werten aus dem Bereich [-1, 1] initialisieren.
Verwenden Sie außerdem je einen Bias in der Eingabe- und der *Hidden*-Schicht.

Es kann hier unter Umständen ein wenig knifflig sein, die richtigen Parameter zum Training des Netzes zu finden. Falls das Ergebnis nicht zufriedenstellend ist, könnte es hilfreich sein, diese Parameter etwas zu variieren.

nach oben

Hinweise zu Aufgabe 6.3

Nutzen Sie die Informationen über die Beschaffenheit der Signale, um Ihre Trainingsmenge aufzubauen und damit Ihr Netz zu trainieren.

Zum Wiederherstellen der Rechtecksignale verwenden Sie dann die Gewichtsmatrizen mit dem geringsten Fehler bezüglich der Validierungsmenge. Dies ist eine effektive Möglichkeit, das Overfitting-Problem zu vermeiden und die Generalisierungsfähigkeiten des Netzes zu erhalten.

Die Signale in den vier Dateien sind recht lang (1024 Datenpunkte), um sie komplett einem Multilayer-Perzeptron als Eingabe zu übergeben. Das Training eines Netzes mit so vielen Eingaben dauert eine Weile. Sinnvoller wäre es, ein Netz zu trainieren, das kleinere Abschnitte (etwa 256 Datenpunkte) verarbeitet. Nach abgeschlossenem Training können die Signale der vier Dateien dann stückweise durch das MLP geschickt werden, um die Rechtecksignale wiederherzustellen.

Die verrauschten Rechtecksignale lassen sich fast perfekt wiederherstellen. Sollte Ihr Netz dennoch sehr viele Fehler machen, dann könnte es hilfreich sein, die freien Parameter ihres Netzes anzupassen. (Neuronenanzahl, nichtlineare Ausgabefunktion im *hidden layer*, abgewandelte Trainingsmenge, andere Lernrate,...)

Das Verfahren, ein Netz aus mehreren gleichartig trainierten Teilnetzen zusammenzusetzen, deren Ausgaben man dann kombiniert, bezeichnet man auch als *ensemble averaging*. Spezialisierte Netze, die für sich genommen nur eine mittelmäßige Performance besitzen, können auf diese Weise zu einem verbesserten System zusammengesetzt werden, das einen kleineren Fehler produziert, als dies jede der Teilkomponenten für sich tun würde. Dieser Aufgabenteil demonstriert diesen Sachverhalt.

Für den Fall, dass Sie das Signal ohne Rechtecksignal nicht finden

können und für jede Datei anscheinend "echte Rechtecksignale" bei der Rekonstruktion erhalten, könnte eine leicht modizierte Trainingsmenge Abhilfe schaffen.

nach oben

Anmerkungen zu Aufgabe 6.4

AdaLinE wird hier sehr viele Fehler machen, seien Sie also deswegen nicht allzusehr beunruhigt. Das liegt daran, daß AdaLinE hier eine *lineare* Prädiktion vornimmt, d.h. der nächste Wert wird als eine gewichtete Summe der vorhergehenden Werte modelliert. Dieses Modell trifft auf die vorliegende Folge allerdings nicht zu, die Abhängigkeiten sind hier stark nichtlinear.

Sinn und Zweck hierbei ist es, zu demonstrieren, daß man mit MLPs ein recht starkes Werkzeug in der Hand hält, das in der Lage ist, diese Nichtlinearitäten zu lernen. Eine genaue Prädiktion von Folgengliedern durchführen zu können, ohne die zugrundeliegende Struktur der Folge kennen zu müssen, ist eine äußerst nützliche Fähigkeit. Ein mögliches Anwendungsgebiet ist etwa die verlustfreie Datenkompression von Bildern, Texten, oder Audiodaten.

Details über Prädiktion zur Datenkompression finden sich etwa im Buch von David Salomon, "Data Compression - The Complete Reference".