Computational Intelligence

M. Sc. Andreas Buschermöhle, M. Sc. Jan H. Schoenke

Übungsblatt 6 - Neuronale Netze II: Lernverfahren

Abgabe bis Mittwoch, 18.06.2014, 12:00 Uhr

In diesem Übungsblatt werden Algorithmen in Matlab geschrieben und zur Auswertung ein- und zweidimensionale Plots angefertigt. Erstellen Sie im Dateiordner Ihrer Gruppe in Stud.IP einen Ordner mit dem Namen `Blatt 6` und laden Sie zu zweit sowohl die m-Files, als auch die fertigen Matlab-Figures mit sinnvoller Achsenbeschriftung rechtzeitig in diesen Ordner hoch. Geben Sie in Vips je nach Aufgabenstellung die textuelle Interpretationen Ihrer Ergebnisse sowie ggf. die Namen der Dateien an, die sich auf die jeweilige Aufgabe beziehen.

Aufgabe 6.1: Lernverfahren für MLPs (20 + 20 = 40 P)

In Matlab stehen bereits verschiedene Lernverfahren zur Verfügung. Diese sollen im Folgenden gegeneinander in zwei Lernszenarien getestet werden. Zum Vergleich der Verfahren soll jeweils ein Netz erstellt werden, dass für jedes verwendete Verfahren kopiert wird. Zum Training werden auch für jedes Verfahren die gleichen Daten verwendet. Auf den Netz-Kopien werden dann die elf Verfahren `traingd`, `traingdm`, `traingdx`, `trainrp`, `traincgf`, `traincgp`, `traincgb`, `trainscg`, `trainbfg`, `trainoss`, `trainlm` zum Training angewandt. Speichern Sie für jedes Verfahren den Verlauf des Testfehlers über die Epochen und tragen Sie diese gemeinsam in einem Plot auf.

a) Ein einfaches Beispiel ist der bereits in den vergangenen Blättern verwendete Cosinus auf dem Wertebereich von $[-2\pi, 2\pi]$. Dieser soll mit einem Netz mit einer verdeckten Schicht mit 6 Neuronen trainiert werden. Verwenden Sie für das Training 20, für die Validierung 10 und für den Test 100 zufällige Daten. Interpretieren Sie Ihre Ergebnisse.

b) Eine komplexere Funktion mit zwei Eingaben ist

$$f(x, y) = sin(x) \cdot (0.01 \cdot y^4 + 0.3 \cdot y^3 - 2 \cdot y^2 + y)$$

auf dem Wertebereich $[0, 4\pi]$ für x und $[0, 5]$ für y. Diese Funktion soll mit einem Netz mit drei verdeckten Schichten mit je 5 Neuronen in den ersten beiden und 2 Neuronen in der dritten Schicht trainiert werden. Verwenden Sie für das Training 500, für die Validierung 100 und für den Test 1000 zufällige Daten. Interpretieren Sie Ihre Ergebnisse.

Aufgabe 6.2: Verhinderung von Overfitting (15 + 15 = 30 P)

Um Overfitting zu verhindern, lassen sich drei verschiedene Methoden anwenden. In den vergangenen Blättern wurde bereits Early Stopping angewendet, um den Trainingsvorgang abzubrechen, sobald auf einer Validierungsmenge der Fehler ansteigt. Alternativ lassen sich durch Regularisierung Gewichte mit kleinen Werten bevorzugen und so der Lösungsraum einschränken. Eine weitere Möglichkeit besteht darin, durch Pruning das Netz soweit zu verkleinern, dass die Ausdrucksstärke nicht mehr zu groß ist und es somit nicht zu Overfitting kommen kann. Da Pruning von der Matlab-Toolbox nicht ohne weiteres unterstützt wird, werden in dieser Aufgabe nur die beiden ersten Varianten betrachtet.
Am Beispiel des Cosinus auf dem Wertebereich von $[-2\pi, 2\pi]$ sollen die beiden Varianten verglichen werden. Die Zielfunktion soll mit einem MLP mit einer verdeckten Schicht mit

25 Neuronen approximiert werden. Nutzen Sie zum Training 60 zufällige Trainingsdaten mit dem Levenberg-Marquardt Algorithmus. Zur Überprüfung des Ergebnisses muss jeweils der Fehler (MSE) gegen die Trainingsmenge und gegen eine Testmenge mit 200 linear verteilten Testdaten ermittelt und der resultierende Verlauf geplottet werden. Ermitteln Sie zunächst als Referenz ohne Verhinderung von Overfitting, also ohne Validierungsmenge, das Resultat nach 2000 Epochen.

a) Untersuchen Sie den Einfluss von Early Stopping, indem Sie zusätzlich beim Training eine Validierungsmenge mit 60 linear verteilten Daten verwenden. Wie ändern sich die Fehlerwerte und der resultierende Funktionsverlauf?

b) Untersuchen Sie den Einfluss von Regularisierung, indem Sie die Performance Funktion (`performFcn` und `performParam.ratio`) entsprechend einstellen. Wie ändern sich die Fehlerwerte und der resultierende Funktionsverlauf? Ist eine Abhängigkeit von dem einstellbaren Parameter `performParam.ratio` zu erkennen?

Aufgabe 6.3: Autorennen 2.0 (30 P)

In dieser Aufgabe soll das Autorennen von Blatt 2 wiederbelebt werden. Die dort evolutionär entwickelte Steuerung wird hier durch ein Neuronales Netz realisiert. Dieses soll mit gegebenen Trainingsdaten trainiert werden, um so von den Trainingsdaten zu verallgemeinern und eine funktionierende Steuerung zu entwickeln. Für die Lenk- und Beschleunigungsaktion muss jeweils ein eigenes MLP erstellt und trainiert werden. Hierzu sind in der Datei `traindata.mat` in Stud.IP 10000 Trainingsdaten zeilenweise in folgender Reihenfolge gegeben: `dx`, `dy`, ϕ, Beschleunigung, Lenkung. Die Eingabe des MLP besteht jeweils aus einem Biasneuron, der Entfernung zum Ziel in x-Richtung (`dx`), der Entfernung zum Ziel in y-Richtung (`dy`) und der aktuellen Ausrichtung des Fahrzeugs (ϕ).
Das Trainieren eines MLPs für die Steuerung des Autos ist nicht einfach möglich. Dies liegt daran, dass die Zielfunktion Unstetigkeiten enthält und diese mit einer verdeckten Schicht nur mäßig approximiert werden können. Daher soll in dieser Aufgabe die k-fache Kreuzvalidierung genutzt werden, um die angemessene Netztopologie für das Problem zu identifizieren. Teilen Sie die 10000 Trainingsdaten in 10 gleichgroße Datensätze (k=10) auf und führen Sie mit diesen die k-fache Kreuzvalidierung für MLPs mit 2 Schichten mit jeweils 1 bis 10 Neuronen durch. Tragen Sie die Ergebnisse in einer Tabelle für die verschiedenen Topologien ein. Welche der Topologien ist demzufolge die beste Wahl und wie viele Ziele können damit in 5000 Simulationsschritten erreicht werden?