HS Osnabrück
IuI – Biermann

Praktikum zu Wissensbasierten Systemen

Blatt 4

Vorgesehen für die Praktikumssitzungen am 20. Mai und 24. Mai 2016

1. Aufgabe: Mit Hilfe der Differentialrechnung sollen auf dem Rechteck $R = (-2, 3)^2$ lokale Minima der Funktion

$$\begin{aligned} \mathrm{f}(\vec{x}) &= \mathrm{f}(x_1, x_2) \\ &= ((x_1+1)^2 + (x_2+1)^2) \cdot ((x_1-2)^2 + (x_2-1)^2) + x_1 + x_2 \end{aligned} \tag{0.1}$$

gesucht werden. Verwenden Sie dabei das in der Vorlesung beschriebene Gradientenverfahren; nehmen Sie dabei aber die im Folgenden beschriebene Verfeinerung vor; diese Verfeinerung trägt der Tatsache Rechnung, daß ein Schritt nicht unbedingt zu einer Verminderung des Funktionswertes $\mathrm{f}(\vec{x})$ führen muß, und bewirkt zusätzlich eine Beschleunigung:

(a) Wähle zufällig Anfangswert $\vec{x}_0 \in R$ und Schrittweite $\sigma = \min(1, 1/||\nabla \mathrm{f}(\vec{x}_0)||$. Setze $i = 0$.

(b) Berechne $\vec{x}_{i+1} = \vec{x}_i - \sigma \nabla \mathrm{f}(\vec{x}_i)$.

(c) Führt dieser Schritt zu einer Verkleinerung, d. h. ist $\mathrm{f}(\vec{x}_{i+1}) < \mathrm{f}(\vec{x}_i)$, so vergrößere man die Schrittweite geringfügig: $\sigma = 1.01 \cdot \sigma$. Setze $i = i + 1$.

(d) Führt hingegen dieser Schritt *nicht* zu einer Verkleinerung, d. h. ist $\mathrm{f}(\vec{x}_{i+1}) > \mathrm{f}(\vec{x}_i)$, so nehme man diesen Schritt zurück, vermindere die Schrittweite: $\sigma = \sigma/2$ und fahre mit (b) fort.

(e) Sofern sich $\mathrm{f}(\vec{\mathrm{x}}_{\mathrm{i}})$ und $\mathrm{f}(\vec{x}_{i+1})$ noch um mehr als 0.01 unterscheiden und $i < 100$ ist, fahre man mit (b) fort.

(f) Andernfalls gebe man $\vec{x}_{i+1}$, $\mathrm{f}(\vec{x}_{i+1})$, $\nabla \mathrm{f}(\vec{x}_{i+1})$ und i aus.

Programmieren Sie dieses in *Matlab* als *Matlab*-Prozedur oder als *Matlab*-Funktion[1]; beides ist in eine „.m“-Datei zu stellen. Beachten Sie dabei, daß eine öffentliche *Matlab*-Funktion in eine „.m“-Datei gleichen Namens gestellt werden muß. Beispiel: Zur gemeinsamen Berechnung der Funktion $\mathrm{f}(\vec{x})$ sowie deren Gradienten $\nabla \mathrm{f}(\vec{x})$ kann eine *Matlab*-Funktion folgender Art erstellt werden, die zwei Werte zurückliefert:

```
function [y,dy]=funk1(x1,x2)
y=y=((x1+1).^2+(x2+1).^2).*((x1-2).^2+(x2-1).^2)+x1+x2;
dy= <<Hierhin der Gradient als Spaltenvektor>> ;
```

Dieser Code[2] muß in eine Datei namens „funk1.m“ gestellt werden.

[1] Informieren Sie sich anhand der *Matlab*-Dokumentation und der *Matlab*-Hilfe über die Erstellung benutzereigener *Matlab*-Funktionen sowie über die hierzu in *Matlab* vorhandenen Kontrollstrukturen.

[2] Beachten Sie die beiden Operatoren „. ˆ “ und „.*“; diese dienen zum komponentenweisen Rechnen mit Feldern; hier ermöglichen sie, die Funktion auf die übliche Weise zu plotten.

Veranschaulichen Sie den Ablauf des Algorithmus durch Einzeichnen des Streckenzuges zu den $\vec{x}_1, \vec{x}_2, \vec{x}_3 \ldots$ in ein Höhenliniendiagramm (*Matlab*-Funktion `contour`) der Funktion (0.1).

Bestimmen Sie die Anfangsschätzung $\vec{x}_1$ durch einen Mausklick in das Höhenliniendiagramm (*Matlab*-Funktion `ginput`).

Fertigen Sie von der Funktion (0.1) zusätzlich ein dreidimensionales Diagramm an.

Überlegen Sie sich eine weitere möglichst interessante reelle Funktion zweier Veränderlicher und führen Sie die Minimumssuche für diese Funktion durch.

Hinweis: Diese Aufgabe dient zur Vorbereitung des *backpropagation*-Algorithmus bei Neuronalen Netzen. Ein Teil des bei dieser Aufgabe erstellten Codes wird man an passender Stelle wiederverwenden können.