Protokoll

Klassifikation mit Fehlerkorrektur-Algorithmus in 2D

LV Mustererkennung

Matthias Jauernig (06INM)
FB IMN, HTWK Leipzig

23.01.2008

1 Aufgabenstellung

Als Projekt im Fach Mustererkennung habe ich mich für die Implementation des Fehlerkorrektur-Algorithmus in 2D entschieden. Dabei werden 2 Punktemengen klassifiziert, indem inkrementell eine Gerade gefunden wird, welche die beiden Mengen voneinander trennt. Voraussetzung dabei ist lineare Separierbarkeit. Als zu klassifizierende Objekte wurden Großbuchstaben des lateinischen Alphabets heran gezogen. Diese sollten rotiert, verschoben und skaliert werden können, es sollten zudem 2 Schriftarten verwendet werden können. Die Merkmale waren so zu wählen und zu implementieren, dass eine korrekte Klassifikation unter diesen Transformationen möglich ist. Die Zweidimensionalität erlaubt nur die Nutzung von 2 Merkmalen zur Klassifizierung. Es war zu ermitteln, ob durch die implementierten Merkmale in jedem Fall lineare Separierbarkeit erreicht werden kann.

2 Entwickeltes DIAS-Programm

2.1 Programmoberfläche und Programmablauf

Das entwickelte DIAS-Programm `fehlerkorrektur.amb` benutzt 3 Fenster zur Darstellung der Funktionalität und Abarbeitung. Über die Menüstruktur des Programms lassen sich verschiedene Parameter einstellen. Bei Programmstart wird eine gespeicherte Konfiguration aus der Datei `fkorr.cfg` geladen, sofern diese vorhanden ist. Sollte das nicht der Fall sein, so wird sie mit einer Default-Konfiguration neu angelegt. Sobald der Benutzer einen Konfigurationsparameter modifiziert, wird die Konfiguration erneut in die Datei geschrieben und steht dann bei weiteren Programmstarts zur Verfügung.

Zunächst werden in Fenster 1 das Hauptmenü sowie die Untermenüs angezeigt. Der Benutzer kann per Maus mit diesen Menüs interagieren und so die gewünschten Optionen einstellen. Im Hauptmenü ist es möglich zu konfigurieren, welche Buchstaben klassifiziert werden sollen, welche Merkmale für die Klassifikation zu verwenden sind und wie erzeugte Buchstaben verändert werden können (Rotation, Skalierung, verschiedene Schriftarten), bevor sie klassifiziert werden. Weiterhin kann mit der eingestellten Konfiguration die Verarbeitung gestartet werden. Über den letzten Menüpunkt wird das Programm beendet.

Wird mit der Verarbeitung begonnen, so werden in Fenster 3 zunächst die Punktemengen anhand der eingestellten Parameter erzeugt. Dazu wird der jeweilige Buchstabe gezeichnet, die Kontur berechnet und dann die Merkmalsberechnung durchgeführt. Die Resultat-Punkte werden im Registerfeld gespeichert.

Nach der Erzeugung der Punktemengen werden diese in Fenster 2 dargestellt. Die Punkte des ersten gewählten Zeichens werden als Kreise dargestellt, die Punkte des zweiten gewählten Zeichens als Kreuze. Nun kann die Verarbeitung noch abgebrochen werden oder es wird mit dem Fehlerkorrektur-Algorithmus fortgefahren, wobei eine schrittweise und eine kontinuierliche Abarbeitung möglich ist. Bei schrittweiser Ausführung wird

nach jeder Veränderung der Trenngeraden diese neu gezeichnet und der Programmablauf pausiert, bei kontinuierlicher Abarbeitung wird der Fehlerkorrektur-Algorithmus solange ausgeführt, bis eine Trenngerade gefunden wurde oder die maximale Anzahl an Iterationen erreicht ist. Wurde eine Trenngerade gefunden, so wird diese grün eingezeichnet und das Programm kehrt zum Hauptmenü zurück.

2.2 Implementierte Merkmale

Für die durchgeführten Tests wurden mehrere Merkmale implementiert, die zur Klassifikation heran gezogen werden. Die Merkmalswerte bewegen sich im Bereich $[0 \dots 1]$ und wurden für die Anzeige auf die gewünschte Größe skaliert. Derzeit sind die folgenden Merkmale implementiert:

- **Formfaktor**: „Inverser Formfaktor“, $4 * \text{Regionengröße} * \pi / \text{Konturlänge}^2$.
- **Konvex-Formfaktor**: Mit den Daten der konvexen Kontur erzeugter „inverser“ Formfaktor.
- **Füllungsgrad**: Regionengröße/Größe umschreibendes Rechteck (nicht rotations-invariant).
- **Flächenrelation**: Größe ohne Löcher/Größe mit Löchern.
- **Elongation**: Über `EFEATR` erhält man die große (λ_1) und kleine (λ_2) Achse einer angenäherten Ellipse. Der Merkmalswert ergibt sich mit λ_2/λ_1.
- **Konvexitätsmaß**: Über `CCFEATR` erhaltener Konvexitätswert geteilt durch 1000.
- **Flächenrelation Kontur:Konvexe**: Fläche des Objekts/Fläche der konvexen Hülle.
- **Längenrelation Konvexe:Kontur**: Länge der konvexen Hülle/Länge der Objektkontur.

2.3 Variationen der Zeichen

Um die Klassifikation unter erschwerten Bedingungen zu testen, können die Zeichen zuvor zufällig wie folgt variiert werden:

- **Verschiedene Zeichensätze**: Als Zeichensätze kommen „Arial“ und „Times New Roman“ zum Einsatz.
- **Rotation**: Ein Zeichen kann zufällig im Bereich $[-45° \cdots + 45°]$ rotiert werden.
- **Skalierung**: Die Zeichengröße liegt im Bereich $[32 \dots 200]$.

3 Praktische Probleme

Das in der Vorlesung angegebene Lernverfahren führt nach endlich vielen Schritten zu einem Ergebnis, wenn die beiden Punktemengen linear separierbar sind. Bei der praktischen Abarbeitung dauerte dieser Prozess aber sehr lang, es waren teilweise bis zu 100000 Iterationen notwendig. Das ist mit dem „Lernen“ der Trenngerade zu erklären. Diese ist gegeben durch $g: \ c_0+c_1x+c_2y=0$. Es ergibt sich für den Punkt $y_0 = g(0): \ y_0 = -c_0/c_2$. Ist der Wertebereich von x, y nun viel größer als 1, so wird durch den Lernprozess y_0 in jedem Schritt nur minimal beeinflusst und das Lernen dauert entsprechend lange, wenn die Trenngerade ein $|y_0| \gg 0$ besitzt. Als Modifikation des Lernverfahrens wird im entwickelten Programm statt $c_0[i+1] = c_0[i]+\alpha$ die Vorschrift $c_0[i+1] = c_0[i]+\gamma\cdot\alpha$ verwendet. Dabei stellte sich ein $\gamma \approx 5 \cdot (x_{max} \cdot y_{max}/2)$ als gut heraus, um ein schnelles Lernen zu ermöglichen.

Weiterhin stellte sich bei Verwendung von γ heraus, dass es unvorteilhaft ist α nach dem Algorithmus der Vorlesung „optimal“ wählen zu lassen. Dadurch wird der gerade betrachtete Punkt zwar im aktuellen Schritt der richtigen Punktemenge zugeordnet, allerdings stagniert das Verfahren, da mit der Zeit kleinere α gewählt werden und y_0 so nicht mehr stark verändert wird. Daher wird in der jetzigen Implementation eine Lernrate verwendet, die mit einem α_0 beginnt und in jeder Iteration den Wert $0.999 \cdot \alpha$ erhält, bis $\alpha < 1$.

4 Experimentelle Auswertungen

Es wurden verschiedene experimentelle Auswertungen zwischen je zwei Großbuchstaben durchgeführt, die hier kurz angerissen werden. Dabei wurden alle Variationen der Zeichen aktiviert. Es wurde festgestellt, dass sich fast alle untersuchten Kombinationen mit 2 Merkmalen trennen lassen, z. B.: (A,V), (B,D), (B,P), (B,S), (C,G), (C,O), (O,Q), (E,F), (E,H), (E,M), (F,K), (K,L), (M,N), (M,X), (M,Y), (I,J), (J,L), (P,R), (U,V), (U,W), (T,Y), (X,Y), (X,Z). Es mussten allerdings teilweise unterschiedliche Merkmale verwendet werden. D. h. dass es keine Kombination von 2 der implementierten Merkmale gibt, durch die alle Buchstabenpaare getrennt werden konnten. Dies impliziert, dass der 2D-Merkmalsraum in der Dimensionierung zu klein war und mind. 3 Merkmale für eine Trennung aller Buchstaben erforderlich sind.

Eine Ausnahme von der linearen Trennbarkeit stellt das Buchstabenpaar (I,L) dar. Es wurden alle Kombinationen von Merkmalen untersucht, dieses Paar lässt sich allerdings durch keine 2 Merkmale linear separieren. Erst wenn nur eine Schriftart Verwendung findet, können die Punktemengen durch eine Gerade voneinander getrennt werden. Es ist möglich, dass dies bei weiteren nicht untersuchten Buchstabenpaaren der Fall ist.

Die einzelnen Auswertungen können weiter bei der Endabgabe des Programms gezeigt und diskutiert werden.