Computational Intelligence

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

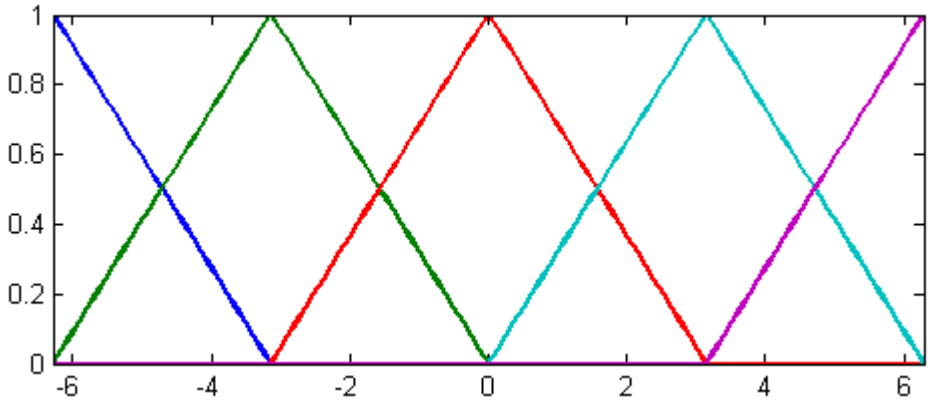
Übungsblatt 10 - On-line Lernen: PA, RLS & UOSLib

Abgabe bis Mittwoch, 16.07.2014, 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 10` 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 10.1: Passive-Aggressive (10 + 5 + 10 + 15 = 40 P)

In dieser Aufgabe soll der Passive-Aggressive-Algorithmus für LIP-Approximatoren umgesetzt und untersucht werden. Als Zielfunktion dient der Cosinus auf dem Wertebereich von $[-2\pi, 2\pi]$. Als LIP-Approximator soll ein Takagi-Sugeno-Fuzzy-System 0. Ordnung mit sum-prod-Inferenz und dreieckigen Zugehörigkeitsfunktionen verwendet werden. Die Zentren der Zugehörigkeitsfunktionen sollen dabei gleichmäßig über dem Wertebereich verteilt werden. Die folgende Abbildung zeigt beispielhaft den Verlauf von fünf Zugehörigkeitsfunktionen über dem Wertebereich:



Die Auswertung von LIP-Approximatoren lässt sich als Skalar-Produkt zwischen einem Parametervektor $\vec{\omega}$ und einem Vektor von Basisfunktionen $\vec{\phi}(x)$ darstellen:

$$f(x) = \vec{\omega}^T \cdot \vec{\phi}(x)$$

Die Basisfunktionen $(\phi_1(x), \ldots, \phi_n(x))^T = \vec{\phi}(x)$ entsprechen bei einem TS-Fuzzy-System gerade den Prämissen, also hier im eindimensionalen Fall den Zugehörigkeitsfunktionen. Die Parameter entsprechen den jeweiligen Konklusionen der zugehörigen Regel und geben hier die Höhenwerte an den jeweiligen Positionen der Zentren der Zugehörigkeitsfunktionen an; Zwischen diesen Höhenwerten wird linear Interpoliert.

a) Schreiben Sie Matlab eine Funktion, welche die Auswertung der Basisfunktionen $\vec{\phi}(x)$ für das oben beschriebene TS-Fuzzy-System durchführt:

```
function out = evalPhi(x,N)
```

In `x` werden der Funktion dabei die Punkte als $nrData \times 1$-Matrix übergeben, an denen die Basisfunktionen ausgewertet werden sollen. Über den Parameter `N` wird die Anzahl der Basisfunktionen festgelegt. Die Ausgabe `out` enthält die Auswertung aller Basisfunktionen für alle Eingaben als $nrData \times$`N`-Matrix.

b) Setzen Sie nun den Passive-Aggressive-Algorithmus zum On-line Lernen mit folgendem Funktionsprototypen um:

```
function paramNew = learnPA(x,y,param)
```

Als Eingaben erhält diese Funktion die Position `x` im Eingangsraum an der gelernt werden soll, die Vorgabe der Höhe `y`, die an dieser Position erreicht werden soll, und den aktuellen Parametervektor `param`. Die Ausgabe der Funktion besteht aus dem mit Hilfe des Lerndatums aktualisierten Parametervektor `paramNew`.

c) Betten Sie nun Ihren Lernalgorithmus aus Aufgabenteil b) in ein On-line Lernszenario für den Cosinus auf dem Intervall $[-2\pi, 2\pi]$ ein. Präsentieren Sie dazu dem Algorithmus nacheinander 200 zufällig aus dem Wertebereich gezogene Lerndaten und verwenden Sie zur Approximation $\texttt{N} = 16$ Basisfunktionen sowie einen mit null initialisierten Parametervektor. Untersuchen Sie den Lernverlauf. Bilden Sie dazu für jeden Lernschritt den MSE der aktuellen Ausgabe des Approximators zur Zielfunktion auf einem gleichmäßigen Raster von 100 Punkten über dem Eingabeintervall $[-2\pi, 2\pi]$. Führen Sie mehrere Durchläufe des Experiments durch, um den Einfluss der zufällig gezogenen Trainingsdaten zu beobachten. Wählen Sie aus den verschiedenen Läufen einige repräsentative aus und stellen Sie für diese den Verlauf des MSE über die Anzahl der Trainingsdaten graphisch dar. Interpretieren Sie Ihre Ergebnisse.

d) Im Kontext des vorangegangenen Aufgabenteils soll nun systematisch der Zusammenhang zwischen der Anzahl an Basisfunktionen `N`, der Anzahl der Trainingsdaten und der Qualität der Approximation in Form des MSE untersucht werden. Bilden Sie dazu ein Raster aus Paaren für die Anzahl der Basisfunktionen und Trainingsdaten und bestimmen Sie für jeden Punkt des Rasters den MSE nachdem alle Trainingsdaten dem Lernalgorithmus einmal präsentiert wurden. Sichern Sie Ihre Ergebnisse gegen die zufällige Auswahl der Trainingsdaten indem Sie jeweils den Mittelwert über mehrere Läufe bilden. Stellen Sie Ihre Ergebnisse graphisch dar und interpretieren Sie sie.

Hinweis: Die Auswahl der Anzahl von Basisfunktionen sollte den Bereich zwischen 1 und 100 nicht verlassen. Die Auswahl der Anzahl von Trainingsdaten sollte den Bereich zwischen 1 und 2000 nicht verlassen. Je nach Auflösung des von Ihnen gewählten Rasters kann die Auswertung sehr lange dauern, beginnen Sie also frühzeitig mit der Bearbeitung.

Aufgabe 10.2: UOSLib (10 + 10 + 10 + 10 = 40 P)

In dieser Aufgabe werden verschiedene Lernverfahren aus dem Bereich des On-line Lernens in unterschiedlichen Szenarien untersucht. Die Untersuchungen werden mit Hilfe der UOSLib[1] durchgeführt. Um eine Untersuchung im Sinne der UOSLib zu spezifizieren, muss ein sogenannter *footprint* erstellt werden. Dieser *footprint* enthält alle notwendigen Informationen, um die Untersuchung eindeutig und reproduzierbar zu beschreiben und gliedert sich in vier Teile:

1. Beschreibung des Lernproblems:
   - Regression: Lernen von kontinuierlichen Zusammenhängen $\mathbb{R}^n \to \mathbb{R}$
   - Klassifikation: Lernen von kategorischen Zusammenhängen $\mathbb{R}^n \to [-1, 1]$
2. Beschreibung des Lernszenarios:
   - Zielfunktion nach welcher die Trainingsdaten generiert werden
   - Anzahl der Trainingsdaten
   - Anzahl der *ground truth* Daten
   - Rauschniveau
   - Reihenfolge der Daten
   - Random seed des Zufallszahlengenerators
3. Beschreibung des Approximators:
   - Globale oder lokale Approximatorstruktur
   - Anzahl der Parameter (Ausdrucksstärke)
   - Vorinitialisierung
   - Bei lokalen Approximatorstrukturen: Art der Interpolation
4. Beschreibung des Lernverfahrens:
   - Auswahl eines Lernverfahrens
   - Parametrierung des Lernverfahrens

Der Aufruf der UOSLib zur Durchführung einer Untersuchung erfolgt über folgende Methode:

```
icl_base(mode, learnMethod, algSetup, model, start,
         targetFunc, livePlot, fastmode, quiet, rSeed)
```

Die verschiedenen Parameter spiegeln dabei zum Einen die Informationen aus dem *footprint* wider, zum Anderen können weitere Parameter angegeben werden, die die genaue Ausführung der Untersuchung zum Beispiel hinsichtlich der Visualisierung betreffen. Eine genaue Beschreibung der einzelnen Parameter ist in der Dokumentation zur UOSLib zu finden.
Als Ergebnis einer Untersuchung liefert die UOSLib drei Maße über den zeitlichen Verlauf der Untersuchung: CL (Cumulative Loss) , DL (Data Loss) und GL (Ground truth Loss). Diese drei Maßzahl kennzeichnen unterschiedliche Aspekte, die beim On-line Lernen von Bedeutung sind und ermöglichen so im Zusammenspiel eine aussagekräftige Beurteilung der Leistungsfähigkeit eines Lernverfahrens in einem bestimmten Lernszenario. Das Maß CL bemisst dabei im Wesentlichen die Güte der Vorhersage des Approximators für bisher unbekannte (also nicht zum Training verwendete) Daten, indem der quadratische Vorhersagefehler aufsummiert wird. DL bemisst wie gut sich der Approximator "erinnert", indem der MSE

[1] http://uoslib.cs.uos.de/index.html

der Ausgabe des Approximators zu allen bisher präsentierten Trainingsdaten bestimmt wird und GL bemisst (wie in Aufgabe 10.1) wie gut der Approximator den zugrundeliegenden Zusammenhang der Daten beschreibt, indem der MSE zu einer festen Auswahl von *ground truth* Daten gebildet wird. Eine genaue Beschreibung der einzelnen Maße und deren Berechnung ist in der Dokumentation zur UOSLib zu finden.
Im Rahmen der folgenden Untersuchungen werden nur Regressionsaufgaben (`mode=1`) betrachtet, außerdem werden nur die Lernverfahren `'PA'` (Passive Aggressive), `'RLS'` (Recursive Least Squares) und `'IRMA'` (Incremental Risk Minimization Algorithm) verglichen. Der `livePlot` wird für eine schnellere Berechnung der Ergebnisse ausgeschaltet, der `fastmode` bleibt deaktiviert, um alle drei Maße zu bestimmen und das Flag `quiet` wird aktiviert, um keine unnötigen Ausgaben zu erzeugen. Außerdem bleibt der `rSeed` fest auf 12345 eingestellt. Änderungen an diesen Einstellungen werden ggf. in den einzelnen Aufgabenteilen explizit gefordert.

a) Vergleichen Sie die drei Lernverfahren in folgendem Szenario:

   - `targetFunc.target = 'sine'`
   - `targetFunc.ND = '300'`
   - `targetFunc.NG = '100'`
   - `targetFunc.noise = 0.0`
   - `targetFunc.minPath = false`

   und verwenden Sie dabei den folgenden Approximator:

   - `model.kind = 'GLT'`
   - `mode.base = 'gauss'`
   - `model.N = 15`
   - `start = 0.`

   Nutzen Sie für die Parametrierung der Lernverfahren folgende Einstellungen:

   - Passive Aggressive:
     `learnMethod = 'PA'`
     `algSetup.variant = 0`
   - Recursive Least Squares:
     `learnMethod = 'RLS'`
     `algSetup.S = 100`
     `algSetup.g = 1`
   - Incremental Risk Minimization Algorithm:
     `learnMethod = 'IRMA'`
     `algSetup.s = 0.1`
     `algSetup.tau = 0.0`
     `algSetup.variant = 1`
     `algSetup.maxstiff = 10.`

   Variieren Sie das Rauschniveau `targetFunc.noise` im Lernszenario im Bereich $[0, 0.1]$ und stellen Sie ihre Ergebnisse graphisch dar. Wie verhalten sich die drei Verfahren hinsichtlich des Einflusses von verrauschten Trainingsdaten? Welches Grundsätzliche Verhalten der einzelnen Fehlermaße lässt sich beobachten?

b) Wiederholen Sie die Untersuchungen aus Aufgabenteil a) mit folgendem Approximator:

   - `model.kind = 'Poly'`

- `model.N = 15`
- `start = 0`.

Dieser hat die gleiche Ausdrucksstärke wie der in Aufgabenteil a) verwendete Approximator, nutzt jedoch zur Appxorimation ein Polynom und hat somit eine globale Approximationsstruktur. Welche Unterschiede zu den Ergebnissen aus Aufgabenteil a) lassen sich beobachten? Interpretieren Sie Ihre Ergebnisse!

c) Wiederholen Sie die Untersuchungen aus Aufgabenteil a) mit folgendem Szenario:

- `targetFunc.target = 'relearn'`
- `targetFunc.ND = '600'`
- `targetFunc.NG = '100'`
- `targetFunc.noise = 0.0`
- `targetFunc.minPath = false`

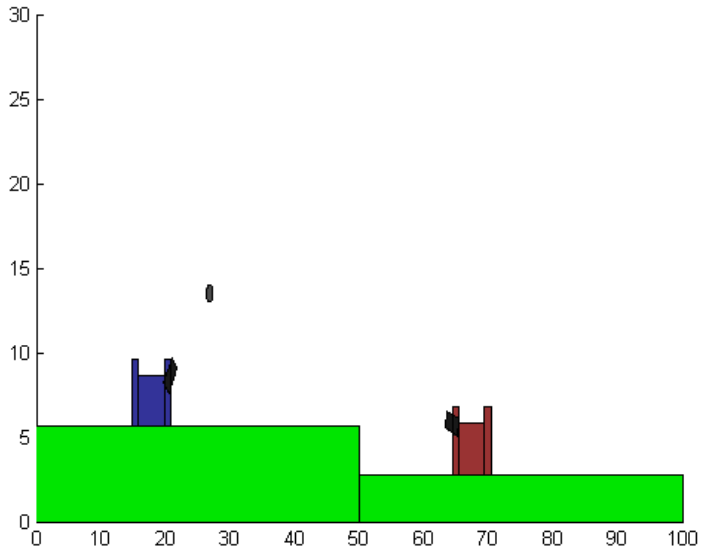
und aktivieren Sie den `fastmode`. Bei diesem Szenario wechselt nach der Hälfte der Lerndaten (hier 300) die Zielfunktion, es handelt sich also um eine nicht statische Zielfunktion, die Sprunghaft ihr Verhalten ändert. Variieren Sie auch hier das Rauschniveau. Stellen Sie Ihre Ergebnisse graphisch dar und interpretieren Sie sie.

d) Wiederholen Sie die Untersuchungen aus Aufgabenteil a) mit folgendem Szenario:

- `targetFunc.target = 'sine'`
- `targetFunc.ND = '300'`
- `targetFunc.NG = '100'`
- `targetFunc.noise = 0.0`
- `targetFunc.minPath = true`.

Betrachten Sie nur den rauschfreien Fall (`targetFunc.noise = 0.0`). Durch das Flag `targetFunc.minPath = true` werden die Trainingsdaten dem Lernalgorithmus entlang einer festen Trajektorie im Eingangsraum präsentiert (andernfalls werden sie zufällig gezogen und bleiben unsortiert, sodass beliebige Sprünge im Eingangsraum möglich sind). Stellen Sie Ihre Ergebnisse graphisch dar und interpretieren Sie sie. Beobachten Sie zusätzlich das Verhalten der einzelnen Verfahren im `livePlot`. Schalten Sie hierfür den `fastmode` ein und `quiet` aus. Welchen Einfluss hat die Trajektorienbehaftung der Trainingsdaten auf die Lernverfahren? Vergleichen Sie Ihre Ergebnisse bezüglich des CL auch mit den Ergebnissen aus Aufgabenteil a).

Aufgabe 10.3: **Grabenkrieg 3.0** **(15 + 5 = 20 P)**



Das aus Aufgabe 7.3 bekannte Szenario wird hier mit einer erweiterten Physiksimulation wiederverwendet, um on-line ein Modell der Physik der Flugbahn der Kugel zu lernen. Dazu sind in Stud.IP die Dateien `Grabenkrieg.m`, `shootBall.m` und `evalShooter.m` gegeben. Die Datei `Grabenkrieg.m` implementiert die Funktion

```
[wins, ILS] = Grabenkrieg(ILS,learnMethod,algSetup,numberOfGames,show,range),
```

welche die Simulation steuert und abwechseln die beiden Spieler Schüsse abfeuern lässt. Wie in Aufgabe 7.3 ist es Ihre Aufgabe ein Modell der Flugbahn der Kugel zu entwickeln und mit Hilfe dieses Modells innerhalb der Funktion `evalShooter` den Abschusswinkel und die Schussstärke der Ihrer nächsten Aktion zu bestimmen. Ihr Modell wird in dieser Aufgabe automatisch in der Funktion `shootBall` mit den jeweils aktuellen Daten der Flugbahn online trainiert. Damit das Modell die Flugbahn korrekt vorhersagen kann müssen Sie zum Einen den Approximator für das Modell spezifizieren und zum Anderen ein Lernverfahren zum Training des Modells auswählen. Verwenden Sie hierzu die Funktionen der UOSLib.
Im Übergabeparameter `ILS` erwartet die Funktion `Grabenkrieg` ein `cell`-Array mit zwei Einträgen. Die beiden Einträge sind die beiden lernfähigen Modelle zur Vorhersage der jeweils nächsten Geschwindigkeit der Kugel in x- und y-Richtung. Als Eingabe erwarten diese Modelle drei Werte: y-Position, x-Geschwindigkeit, y-Geschwindigkeit. Diese Eingaben müssen Sie auch bei Ihren Auswertungen in der Funktion `evalShooter` verwenden. Verwenden Sie für Ihre Simulation innerhalb von `evalShooter` eine Schrittweite von `dt = 0.1`. Über `learnMethod` bestimmen Sie den Lernalgorithmus mit welchem das Modell trainiert wird, es stehen Ihnen alle in der UOSLib implementierten Verfahren zur Verfügung (sofern diese für Regression geeignet sind). Die Parametrierung des von Ihnen gewählten Lernalgorithmus nehmen Sie über `algSetup` vor. Die Anzahl der zu simulierenden Spiele wird über `numberOfGames` festgelegt. Das Flag `show` aktiviert bzw. deaktiviert die Visualisierung. Im Rahmen der UOSLib entstammen alle Eingaben an einen Approximator dem Intervall $[-10, 10]$, um diese Einschränkung aufzuheben können in `range` die Wertebereiche der drei Eingaben als 3×2-Matrix angegeben werden, diese werden dann jeweils linear auf das Intervall $[-10, 10]$ skaliert.
Die in Stud.IP zur Verfügung gestellten Dateien gehen implizit davon aus, dass die UOSLib im Unterordner `src` des Verzeichnisses liegt in dem die Funktion ausgeführt wird, also über den Befehl `addpath('src')`. Stellen sie dies für Ihre Arbeit mit den Dateien sicher.

a) Entwerfen sie lernfähige Modelle für die Vorhersage der Geschwindigkeit in x- und y-Richtung, die als Eingaben die y-Position, x-Geschwindigkeit und y-Geschwindigkeit erwarten und geben Sie die Wertebereiche der jeweiligen Eingaben an, die sie erwarten. Wählen und parametrieren Sie außerdem ein Lernverfahren zum Training der Modelle. Welchen Approximatortyp haben Sie gewählt und wie haben Sie die notwendige Ausdrucksstärke bestimmt. Begründen Sie Ihre Designentscheidungen!

b) Zeigen Sie die Vorhersagegüte Ihres Modells, indem Sie wie in Aufgabe 7.3 mit einer Methode ihrer Wahl mit Hilfe des Modells eine optimale Kombination aus Abschusswinkel und -stärke ermitteln und so den naiven Spieler sicher besiegen. Füllen Sie hierzu die Funktion

```
[angle speed] = evalShooter(ILS, ownPosition, target, range)
```

mit der von Ihnen gewählten Methode sinnvoll aus. Die Funktion erhält als Übergabeargumente das `cell`-Array `ILS`, welches die trainierten Modelle zur Vorhersage der x- und y-Geschwindigkeit enthält, die Position der eigenen Burg in `ownPosition` und die Position der gegnerischen Burg in `target`. In `range` werden die von Ihnen gewählten Wertebereiche der Eingaben übergeben, die Sie zur Skalierung der Eingaben an die Modelle nutzen. Die Einschränkungen bezüglich Abschusswinkel und -geschwindigkeit sind aufgehoben.