183.171
Neural Computation LU
Aufgabenblock 1

Michael Reiter

20. Oktober 2009

Abgabe der Aufgaben 1.1 und 1.2 bis 27.11.2009, 18:00h per email an `nclu@prip.tuwien.ac.at`:

- Lauffähiger MATLAB code in einem zip file (keine Unterverzeichnisse, ein Matlabskript, mit dem alle Ergebnisse erzeugt werden)
- PDF Dokument mit Experimentergebnissen, Dokumentation und ausführlicher Diskussion.
- Alle Fragen müssen in der Dokumentation beantwortet sein.

1 Aufgabe I

Ziel dieses Aufgabenblocks ist es, Erfahrung mit dem Aufbau eines einfachen Perzeptrons und Lernverfahren zu sammeln. Aufbauend darauf sollen Experimente mit einem *Multi Layer Perceptron* durchgeführt werden.

1.1 Teil 1: Einfaches Perzeptron

1.1.1 Datengeneration

Schreiben Sie eine Funktion `[data,target] = genData(n,d)`, die synthetische Daten für ein binäres Klassifikationsproblem generiert. $data$ soll Datenmatrizen, mit n Zeilen (= Anzahl der Beobachtungen) und d Spalten (=Dimension der Eingabevektoren) enthalten. $target$ enthält einen n-Vektor (Zeilenvektor) der zu jeder Beobachtung den entsprechenden Target Wert $\in \{-1, 1\}$ enthält. Sie können obiger Funktion weitere Parameter hinzufügen, mit denen gesteuert werden kann, ob eine linear separierbare oder nicht l.s. Menge generiert werden soll (zB. boolescher Wert oder Mittelwerte, Varianzen der Verteilungen).

Generieren sie damit 2 Datensätze mit jeweils 100 2-dimensionalen Beobachtungen. Die Eingabedaten sollen mit der Matlab-Funktion $randn$ erzeugt werden, wobei

für die 2 Klassen die Eingabevektoren als jeweils 50 Realisationen einer Normalverteilung (d-dimensional) erzeugt werden sollen. Wählen Sie die beiden Mittelwerte und (Ko-) Varianzen für die 2 Klassen so, dass ein linear separierbarer und ein nicht linear separierbarer Datensatz mit jeweils 100 2-dimensionalen Beobachtungen mit entsprechenden Target-Werten entsteht.

Fragen:

- Stellen Sie die Lage der Datenvektoren in $\mathbb{R}^2$ und ihre labels (Targetwerte) graphisch dar.

1.1.2 Perzeptrontraining

Schreiben sie eine Funktion, die ein einfaches Perzeptron simuliert, als Inputwerte den weight-Vektor $\mathbf{w}$ und input Daten $\mathbf{x}$ akzeptiert und den Perzeptronoutput $\mathbf{y}$ liefert, d.h.

```
y = perc(w,x).
```

Implementieren Sie einen Trainingsalgorithmus für dieses Perzeptron d.h.

```
w = percTrain(X,t,maxIts).
```

Input für den Trainingsalgorithmus sind Trainingsdaten $\mathbf{X}$, Targetvektor $\mathbf{t}$ und eine Obergrenze für die Zahl der Trainingsiterationen $maxIts$, output ist ein weight-Vektor $\mathbf{w}$.

Fragen:

- Untersuchen sie den Trainingsalgorithmus: Wieviele Iterationen werden benötigt, bis sich *weight-Vektor* $\mathbf{w}$ im Fall von linear separierbaren Daten nicht mehr ändert?
- Wie verändert sich der $\mathbf{w}$ während des Trainings?
- Welchen Einfluß hat die Schrittweite?
- Plotten Sie Daten und Entscheidungsgrenze in $\mathbb{R}^2$ (analog zu Punkt1.1.1).
- Wie ist das Verhalten bei nicht linear separierbaren Daten?

1.1.3 Perzeptronevaluierung

Verwenden sie für das Training bzw. die Evaluierung Ihre Datensätze und die auf der LU web page für jede Gruppe zur Verfügung gestellten Daten und teilen Sie sie selbstständig in Training- und Testset.

`http://www.prip.tuwien.ac.at/teaching/ws/neural-computation/daten-fur-aufgabe-1`
→ data_GrXX.txt

Fragen:

- Untersuchen Sie die Leistung des trainierten Perzeptrons auf dem Testset abhängig von der Größe des Trainingsets.
- Stellen Sie für einen Vergleich die wahren und die vom Perzeptron generierten Datenlabels graphisch dar.

1.2 Teil 2: Generalisierte lineare Regression - travelling on error surfaces

1.2.1 Funktionsgenerierung

Generieren Sie einen Datenvektor mit $x \in [0, 5]$ (Zeilenvektor mit 1-dimensionalen Eingabewerten) mit $stepsize = 0.1$ und einen entsprechenden Datenvektor $y = 2x^2 - Gx + 1$, wobei $G = Gruppennummer$.

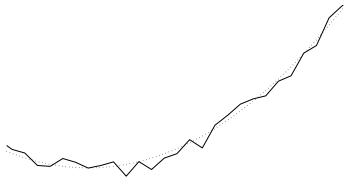

Abbildung 1: Beispiel einer gestörten Funktion (durchgehende Linie) mit dem Ergebniss der Regression (gestrichelte Linie)

1.2.2 Lernen einer Funktion mit quadratischer Regression

Verwenden Sie eine lineare Einheit (LMS Lernregel) um lineare Regression durchzuführen. Führen Sie die Regression auf transformierten Eingabedaten durch. Verwenden Sie beispielsweise folgende Merkmalstransformation: $\mathbf{\Phi}(x) \rightarrow (1, x, x^2)^T$ (Matlab kann elementweise potenzieren: z.B. `[x x x].^ [0 1 2]`). Hinweis: Es ist hilfreich während des Trainings bereits y und die Vorhersage (Ausgabe der linearen Einheit) o zu plotten bzw. die Veränderung des Gewichtsvektors zu verfolgen um schnell zu sinnvollen Werten für die Lernrate γ zu kommen (mögliche Laufzeiteinbußen).

Fragen:

- Wieviele und welche *basis functions* Φ_i verwenden Sie?
- Welche *cost function* ergibt sich?
- Untersuchen Sie den Einfluß der Schrittweite γ auf das Konvergenzverhalten.
- Welches γ ergibt eine guten tradeoff zwischen Trainingszeit und Konvergenzverhalten? Gibt es ein γ, bei dem $\mathbf{w}$ divergiert
- Wieviele Epochen benötigt der Algorithmus?

- Stören sie das Outputsignal y geringfügig (d.h. ± 0.5) und plotten Sie das gestörte Signal und das Ergebniss der Regression dieses Signals.

1.3 NN-toolbox: einige Matlab-Funktionen / Demos

Im zweiten Übungsblock werden Sie sich mit der NN-toolbox von Matlab beschäftigen, um auch komplexere Aufgaben zu untersuchen. Machen Sie sich mit den Funktionen vertraut. Mit dem Befehl: `help nndemos` können Sie auf Demos zu den verfügbaren Funktionen zugreifen. z.B.:

- `nnd12sd1` bzw. `nnd12sd2`: Steepest descent backpropagation demonstration.
- `nnd4pr`: perceptron training demonstration.
- `newff`: generiert ein feedforward backpropagation network.
- `train`: trainiert nn.
- `sim`: simuliert nn (help network/sim).