183.605
Machine Learning for Visual Computing
Aufgabenblock 1

Michael Reiter

23. Oktober 2013

Abgabe der Aufgaben via TUWEL. Bitte beachten Sie die dort erwähnten Abgabefristen.

- 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 später Experimente mit einem *Linear Basis Function Model* 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).

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 linear separierbarere und nicht linear separierbarere Datensätze mit jeweils 100 2-dimensionalen Beobachtungen mit entsprechenden Target-Werten entstehen. Erzeugen Sie mehrere (mindestens 2) linear separierbare Datensätze mit verschiedenen Mittelwerten und Kovarianzen, sodass der "Abstand" der Klassen variiert (minimaler Abstand zweier Beobachtungen aus unterschiedlichen Klassen).

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 Eingabewerte den Gewichtsvektor $\mathbf{w}$ und Eingabedaten $\mathbf{x}$ akzeptiert und die Perzeptronausgabe $\mathbf{y}$ liefert, d.h.

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

Implementieren Sie sowohl das *online*-Trainingsverfahren als auch das *batch*-Verfahren (siehe Vorlesungsfolien), d.h. `w` soll mit einer entsprechenden Matlab Funktion wie folgt ermittelt werden:

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

Eingabe für den Trainingsalgorithmus sind Trainingsdaten $\mathbf{X}$, Targetvektor $\mathbf{t}$ und eine Obergrenze für die Zahl der Trainingsiterationen `maxIts`. `online` gibt an, ob mittels *online*-Verfahren oder *batch*-Verfahren trainiert wird. Die Rückgabe ist ein weight-Vektor $\mathbf{w}$.

Fragen:

- Untersuchen sie den Trainingsalgorithmus: Welche Eigenschaften der Daten beeinflussen die durchschnittliche Anzahl an Iterationen bis eine Lösung $\mathbf{w}^*$ gefunden wurde?
- 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?

1.2 Teil 2: Lineare Regression

Ziel dieser Aufgabe ist es, Parameteroptimierung auf einer Fehlerfunktion umzusetzen und dabei den Zusammenhang zwischen Modellkomplexität (entspricht in diesem Fall Anzahl der Basisfunktionen) und theoretischem Gesamtfehler zu untersuchen.

1.2.1 Datengenerierung

Generieren Sie einen Vektor mit Eingabewerten $x \in [0, 5]$ (Zeilenvektor mit 1-dimensionalen Eingabewerten) mit einer Schrittweite von 0.1 (51 Eingabewerte) und einen entsprechenden Datenvektor y mit Ausgabewerten $y = 2x^2 - Gx + 1$, wobei $G = Gruppennummer$. Diese 51 Punkte sollen zur Visualisierung (Plotten) der gesuchten Funktion herangezogen werden. Erstellen Sie aus den 51 Punkten eine Trainingsmenge indem Sie jeden 6 Punkt auswählen und jeweils einen mit $\mathcal{N}(\mu = 0, \sigma = 0.7)$ normalverteilten zufälligen Wert zu y_i addieren um verrauschte Targetwerte t_i zu erzeugen. Die so erzeugte Trainingsmenge enthält $N = 8$ Paare von Beobachtungen x_i, t_i.

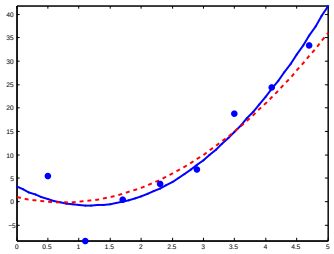

Abbildung 1: Beispiel: Wahre Funktion (rot gestrichelte Kurve), Trainingsmenge mit $N = 8$ (blaue Punkte) und Ergebnis der Regression (blaue Kurve) mit einem Polynom mit Grad 2.

1.2.2 Basisfunktionen und Parameteroptimierung

Verwenden Sie eine lineare Einheit (*online* 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) zu plotten bzw. die Veränderung des Gewichtsvektors zu verfolgen um schnell zu sinnvollen Werten für die Lernrate γ zu kommen (mögliche Laufzeiteinbußen).

Fragen:

- Welchen Gewichtsvektor erhalten Sie mittels Gradientenabstieg auf der quadratischen Fehlerfunktion?
- Wie können Sie den optimalen Gewichtsvektor $\mathbf{w}^*$ im Fall einer quadratischen Fehlerfunktion in einem Schritt berechnen? Vergleichen Sie das so berechnete Optimum mit dem Ergebnis des Gradientenabstiegs.
- 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?

1.2.3 Modellkomplexität und Modellselektion

Ergänzen Sie weitere Basisfunktionen $\mathbf{\Phi}(x) \rightarrow (1, x, x^2, x^3, ..., x^d)^T$ (und vergrößern Sie $\mathbf{w}$ entsprechend). Berechnen Sie $\mathbf{w}^*$ für mindestens 2000 verschiedene Trainingsmengen mit $\sigma = 0.7$. Schätzen Sie den Erwartungswert und die Varianzen der Komponenten (Matlab-Funktion `diag(cov(...))`) von $\mathbf{w}^*$ ($\mathbf{w}^*$ ist eigentlich ein Schätzer der Koeffizienten der wahren Funktion).

Fragen:

- Wie verhält sich der Erwartungswert und wie die Varianzen aller Koeffizienten in $\mathbf{w}^*$ im Bezug zu d? Was können Sie im Speziellen über die Koeffizienten der hinzugefügten Basisfunktionen ($2 < j \leq d$) sagen?
- Plotten Sie für ein fixes x^* (z.B. $x^* = 2$) die mittlere quadratische Abweichung der Modellvorhersage von $f_{\mathbf{w}^*}(x^*)$ vom wahren Funktionswert $f(x^*)$ (anhand der 2000 Trainingsmengen) im Bezug zu $0 \leq d \leq 20$ ($d = 0$ bedeutet konstante Funktion).
- Welches $\mathbf{w}^*$ erhalten Sie bei einer ungestörten Trainingsmenge im Bezug zu d?