Übung zur Vorlesung

Maschinelles Lernen 2

Sommersemester 2007

Abteilung Maschinelles Lernen
Institut für Softwaretechnik und
theoretische Informatik
Fakultät IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: krm@cs.tu-berlin.de

Blatt 1

Abgabe 24. April 2007 in der Vorlesung, oder bis 23:59:59 Uhr bei `mikio@cs.tu-berlin.de`

Die Aufgaben können auch in Gruppen bearbeitet werden, allerdings sollte die Gruppenzusammenstellung über des Semster stabil bleiben. Für praktische Aufgaben bitte ebenfalls Code und Ausgabe (d.h. Ergebnisse, Plots) abgeben. Verwende Matlab oder Octave. Lehrversionen von Matlab sind im cs-Netz (IRB) installiert. Zum Start `ml/bin/matlab` aufrufen. Octave ist ein freie Matlabclone, der unter `www.octave.org` verfügbar ist. Für die Abgabe von praktischen Aufgaben bitte die Coding-Richtlinien beachten, die für das Praktikum gelten (siehe hierzu `http://ml.cs.tu-berlin.de/de/ss07_ml_praktikum.html`, Hinweise auf dem ersten Übungsblatt).

Aufgaben

In der Vorlesung wurde der AdaBoost-Algorithmus vorgestellt. Weiterhin wurden die Verbindungen zwischen Boostingverfahren und dem Gradientenabstieg illustriert. Beide Methoden findet man am Ende dieses Übungsblatts.

1. **Gradientenabstieg (10 Punkte)** Zeige, dass AdaBoost äquivalent zum Gradientenabstieg für die exponentielle Verlustfunktion

   $$L(y, y') \quad = \quad \exp(-yy') \,.$$

   ist, wenn man als schwachen Lernen den Minimierer der quadratischen Verlustfunktion

   $$L_2(y, y'y) \quad = \quad (y - y')^2$$

   verwendet. Gehe dabei folgendermaßen vor.

   (a) Zeige, dass der negative Gradient u_i aus Gleichung ((2), siehe Rückseite) die folgenden Gleichung erfüllt:

   $$u_i \quad = \quad y_i \cdot D_m(\boldsymbol{x}_i)$$

   Dabei sind $D_m(\boldsymbol{x}_i)$ die Gewichte von AdaBoost.

   (b) Setze u_i in die quadratische Verlustfunktion L_2 ein und zeige, dass für eine vorgegebene Schrittlänge $\alpha > 0$ der Minimierer h_m des Verlustes

   $$\sum_{i=1}^{n} L_2 \left(u_i, \alpha h_m(x_i)\right)$$

   gerade der Minimierer des gewichteten 0/1-Fehlers (1) ist.

   (c) Setze die optimale Lösung h_m in Gleichung (3) ein und zeige, dass die optimale Schrittlänge der Formel für α_m aus dem AdaBoost-Algorithmus entspricht.

2. **Implemetierung (10 Punkte)** Implementiere den AdaBoost-Algorithmus. Verwende als schwachen Lerner *stumps*, d.h. achsenparallele Trennebenen. Wende AdaBoost (mit $M = 100$ Iterationen) auf den zweidimensionalen `banana`-Datensatz an, der auf der Vorlesungsseite heruntergeladen werden kann. Plotte die Daten, den Trainingsfehler gegen die Iteration, und zeichne die Entscheidungsgrenze ein.

3. **AdaBoost-Variation (10 Punkte)** Implementiere die folgende Variante von Adaboost: Wähle in jedem Schritt die Gewichte $D_m(\boldsymbol{x}_i)$ zufällig (gleichverteilt). Wende diese Variante ebenfalls auf den `banana`-Datensatz an. Plotte wieder Daten und zeichne die Entscheidungsgrenze ein. Vergleiche das Ergebnis mit AdaBoost.

Algorithm 1 AdaBoost

Eingabe: Daten $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, Anzahl der Boosting-Schritte M, Start-Gewichte $D_1(\boldsymbol{x}_i) = 1/n$

for $m = 1, \ldots, M$ **do**

Wende einen schwachen Lerner h_m auf die gewichteten Daten (S, D_m) an

Berechne den gewichteten Fehler

$$\epsilon_m \quad = \quad \sum_{i=1}^{n} D_m(x_i) I_{\{y_i \neq h_m(x_i)\}} \,. \tag{1}$$

Setze

$$\alpha_m \quad = \ln \quad \left(\frac{1 - \epsilon_m}{\epsilon_m} \right) .$$

Aktualisiere die Gewichte

$$D_{m+1}(x_i) = D_m(x_i) \exp\left(-\alpha_m y_i h_m(x_i)\right) .$$

end for

return $H_M(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right)$.

Algorithm 2 Gradientenabstieg

Eingabe: Verlustfunktion $L(y, y')$, Anzahl der Iterationen M

Wende einen schwachen Lerner h_1 auf die Daten an und setze $f_1 = h_1$

for $m = 2, \ldots, M$ **do**

Berechne den negativen Gradienten der Verlustfunktion

$$u_i \quad = \quad - \left. \frac{\partial L(y, f)}{\partial f} \right|_{f = f_m(x_i)} , i = 1, \ldots, n \,. \tag{2}$$

Wende einen schwachen Lerner h_m auf die **modifizierten** Daten $(\boldsymbol{x}_i, u_i)$ an

Bestimme die optimale Schrittlänge

$$\alpha_m \quad = \quad \arg\min_{\alpha} \left\{ \sum_{i=1}^{n} L\left(y_i, f_{m-1}(\boldsymbol{x}_i) + \alpha h_m(\boldsymbol{x}_i)\right) \right\} . \tag{3}$$

Aktualisiere

$$f_m = f_{m-1}(\boldsymbol{x}) + \alpha_m h_m(\boldsymbol{x}) \,,$$

end for

return f_m

Hinweise

- Zum Lernen der Stumps kann man entweder den quadratischen Fehler minimieren, oder alternativ alle möglichen Schrittweiten für alle Dimensionen ausprobieren und die Trennebene mit dem kleinsten Fehler wählen.
- Zur Darstellung der gelernten Trennebene muß der gelernte Prediktor auf einem Grid ausgewertet werden, und dann die Trennebene zum Beispiel mit `contour` geplottet werden.