Übung zur Vorlesung

Maschinelles Lernen 2

Sommersemester 2008

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

Blatt 3

Abgabe: 6. Mai 2008, bis 12.15 h bei Nicole Krämer (nkraemer@cs.tu-berlin.de)

Für praktische Aufgaben bitte ebenfalls den Code abgeben. Verwende Matlab oder Octave, welches unter www.octave.org frei verfügbar ist. Als Abgabe wird nur die Datei knngraph.m akzeptiert. Insbesondere werden keine zusätzlichen Dateien mit Hilfsfunktionen oder komprimierte Dateien akzeptiert.

Aufgaben

1. **k-nächster-Nachbar-Graph (10 Punkte)** Im halbüberwachten Lernen regularisiert man die Verlustfunktion von Lernalgorithmen, in dem man fordert, dass Punkte mit geringem Abstand möglichst die gleichen Labels haben sollten. Dies kann zum Beispiel über den k-nächsten-Nachbar-Graph geschehen. Dabei repräsentieren die Beispiele $x_1, \ldots, x_n$ die Knoten des Graphen, und eine Kante zwischen x_i und x_j symbolisiert, dass x_i einer der k nächsten Punkte von x_j ist oder dass x_j einer der k nächsten Punkte von x_i ist. Für diese Aufgabe wählen wir als Abstandsmaß

$$\Delta_\sigma(x,z) \quad = \quad 1 - \exp\left(-\frac{\|x-z\|^2}{\sigma^2}\right). \tag{1}$$

   (a) Seit $X \in \mathbb{R}^{d\times n}$ die Matrix, in deren Spalten die Beispiele stehen. Schreibe die Matrix

$$D \quad = \quad \left(\|x_i - x_j\|^2\right) \in \mathbb{R}^{n\times n}$$

   in Abhängigkeit von X. Verwende keine Summenzeichen, und benutze nur Matrix- und Vektormultiplikationen.

   (b) Schreibe eine Funktion

```
A=knngraph(X,k,sigma)
```

   Dabei ist `X` $\in \mathbb{R}^{d\times n}$, `k`$\geq 1$ eine natürliche Zahl und `sigma`> 0 der Parameter des Abstandsmaßes (1). Ausgegeben wird die Adjazenzmatrix `A`$\in \{0,1\}^{n\times n}$ des k-nächsten-Nachbar-Graphen. Verwende keine Schleifen!
   Hinweis: Die Funktion `sort()` gibt nicht nur die sortierten Daten aus, sondern auch die Permutation, die die Daten in die sortierten Daten überführt.

2. **Kerne für Wörter (10 Punkte)** Der "Bag-of-Words"-Kern ist definiert als

$$k_1(x,z) = \sum_{w\in L} \#_w(x)\#_w(z)\cdot N_w,$$

   wobei $\#_w(x)$ die Häufigkeit von w in der Sequenz x ist und die Einbettungssprache L einer natürlichen Sprache entspricht, z.B. Deutsch. Um den Einfluß verschiedener Worte zu Gewichten kann jedem Wort w ein Gewicht N_w zugewiesen werden.

   Häufig wird die inverse Dokumentfrequenz (IDF) als Gewichtung verwendet. Für einen Datensatz D aus Dokumenten bestimmt man für jedes Wort w in D die Teilmenge D_w von Dokumenten, die es enthalten. Die Gewichtung ist dann definiert als

$$N_w = \log|D| - \log|D_w|.$$

   Beweise, dass für diese Gewichtung k_1 tatsächlich ein Kernel ist.

3. **Kerne für N-gramme (10 Punkte)** Der Spektrum-Kern ist ohne Gewichtung definiert als

$$k_2(x, z) = \sum_{w \in L} \#_w(x) \#_w(z) \quad \text{mit} \quad L = \mathcal{A}^n,$$

wobei $\mathcal{A}^n$ die Menge aller Sequenzen der Länge n (N-gramme) ist. Definiert man die Einbettungssprache L als Vereinigung von N-grammen verschiedener Längen, erhält man einen "geblendeten" Spektrum-Kern

$$k_3(x, z) = \sum_{w \in L} \#_w(x) \#_w(z) \quad \text{mit} \quad L = \bigcup_{i=1}^{n} \mathcal{A}^i.$$

Berechne für beide Kerne k_2 und k_3 Kernmatrizen für $n = 3$ über die folgende Menge von Sequenzen:

"ananas", "anna", "natter", "otter", "otto"

4. **Kerne für Teilsequenzen (10 Punkte)** Zur Berechnung des Subsequenz-Kerns wird ein Suffixbaum verwendet.

Zeige, dass ...

(a) eine Sequenz x genau $\frac{|x|^2+|x|}{2}$ Teilsequenzen (Substrings) enthält,

(b) jede Teilsequenz w in $\mathcal{O}(|w|)$ in einem Suffixbaum zu erreichen ist und

(c) ein Suffixbaum nur $\mathcal{O}(|x|)$ Speicher benötigt.