Computational Learning Theory

Sommersemester 2018

Mario Holldack, M. Sc.
Prof. Dr. Georg Schnitger
Hannes Seiwert, M. Sc.

GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Institut für Informatik
AG Theoretische Informatik

Übungsblatt 9

Ausgabe: 11.06.2018
Abgabe: 18.06.2018

Für dieses Übungsblatt benötigen Sie die Python-Bibliothek scikit-learn (und ihre Abhängigkeiten wie etwa NumPy). Empfehlenswert bei der Entwicklung ist die Benutzung der Notebook-Funktion von jupyter.

- http://scikit-learn.org/stable/install.html
- https://www.scipy.org/scipylib/download.html
- https://jupyter.org/install.html

Diese und viele weitere Bibliotheken aus dem Bereich Machine Learning und Data Science finden Sie auch in der Python-Distribution Anaconda:

- https://www.anaconda.com/download/

Reichen Sie Ihren kommentierten Quellcode als *.py- oder *.ipynb-Datei sowie und die Dokumentation ihrer experimentellen Ergebnisse per E-Mail ein: holldack@thi.cs.uni-frankfurt.de.

Aufgabe 9.1 *Ziffernerkennung mit SVMs und scikit-learn* (32 Punkte)

In dieser Aufgabe lernen wir Klassifikationen von Ziffern aus dem MNIST-Datensatz mithilfe von SVMs. Bei MNIST handelt es sich um einen Datensatz von 1797 handgeschriebenen Ziffern aus je 8×8 Pixeln. Laden Sie die unter

http://thi.cs.uni-frankfurt.de/lehre/clt/sose18/clt_sose18_blatt09_daten.zip

bereitgestellte Zip-Datei mit der Jupyter-Notebook-Vorlage `clt_sose18_blatt09_vorlage.ipynb` herunter und lesen Sie die Datei readme.txt:

a) Implementieren Sie jeweils eine Funktion zur Erstellung der folgenden Feature-Vektoren:
   i) 8×8 Pixel und Graustufen.
   ii) Graustufen-Histogramm mit vier Klassen ($[0 \dots 3]$, $[4 \dots 7]$, $[8 \dots 11]$ und $[12 \dots 15]$); der Histogramm-Vektor gibt in der i-ten Komponente an, wie viele Pixel der Grafik einen Grauwert im Intervall $[4i \dots 4i + 3]$ besitzen.
   iii) Ein nicht-trivialer Feature-Vektor Ihrer Wahl[1].

b) Wir wollen eine SVM für das binäre Klassifikationsproblem „erkenne die Ziffer 7“ trainieren. Verwenden Sie hierfür die Python-Klasse SVC (siehe http://scikit-learn.org/stable/modules/svm.html) für jede Wahl von Feature-Funktionen aus a) mit

   i) linearen Kernen $K(x, y) = \langle x, y \rangle$.
   ii) polynomiellen Kernen $K(x, y) = (\langle x, y \rangle + c)^d$ für $d \in \{2, 3\}$ und
   iii) Gauß-Kernen $K(x, y) = \exp(-\frac{\|x-y\|^2}{\sigma^2})$

   für eine geeignete Auswahl von Parametern c und σ. Eine systematische Parameterwahl mittels Kreuzvalidierung ist in dieser Übungsaufgabe nicht nötig. Bewerten Sie jeweils den Trainingserfolg in Abhängigkeit von der Anzahl der Supportvektoren und im Fall der linearen Kerne

[1] Seien Sie kreativ!

auch in Abhängigkeit von der Norm $\|w\|_2$ des Gewichtsvektors (bei linearer Trennbarkeit ist der Margin $1/\|w\|_2$). Gehen Sie dabei auch auf die jeweilige Feature-Funktion ein.

Hinweis: Wandeln Sie die im MNIST-Datensatz gegebenen Klassifikationen $(0, \ldots, 9)$ zunächst in binäre Klassifikationen um.

c) Klassifizieren Sie die Ziffern aus der Testmenge mit der „erfolgreichsten“ SVM aus den vorherigen Teilaufgaben bzgl. der Anzahl der Supportvektoren.

   i) Wie hoch ist der Anteil der positiven bzw. negativen Fehlklassifikationen? Wie hoch ist der Anteil der Fehlklassifikationen (d. h. der Testfehler) insgesamt?

   ii) Wie gut schneidet diese SVM ab, wenn Sie die Anzahl der Trainingsbeispiele reduzieren?

   iii) Die Python-Klasse SVC verfügt über einen Strafparameter $C > 0$, mit dem Sie steuern können, wie sehr Fehlklassifikationen in der Trainingsphase bestraft werden. Je größer C ist, desto höher ist die Strafe.

   Wie verhält sich der Testfehler in Abhängigkeit von C? Wie verändert sich $\|w\|_2$ im Fall linearer Kerne? Begründen Sie Ihre Antwort qualitativ und quantitativ.

Hinweis: Dokumentieren Sie Ihre Experimente und strukturieren Sie den dazugehörigen Python-Code so, dass man einzelne Experimente leicht wiederholen kann. Im Jupyter-Notebook können Sie für die Dokumentation Zellen des Typs `MarkDown` verwenden (siehe auch https://de.wikipedia.org/wiki/Markdown).

Aufgabe 9.2 *Nichtbinäre Klassifikationsprobleme mit SVMs* (4* Extrapunkte)

Beschreiben Sie einen Algorithmus, der mithilfe von (möglicherweise mehreren) SVMs mehr als zwei Klassen lernt und dabei möglichst gut abschneidet. Insbesondere soll Ihr Algorithmus tolerant gegenüber binären Fehlklassifikationen sein.

Eine Implementierung wird hier nicht verlangt.

Hinweis: Welche Mittel aus der Vorlesung sind hierfür hilfreich? Alternativ: Wie wird der „Multiclass“-Fall in scikit-learn implementiert?