Lehrstuhl für Software Engineering

Übungsblatt: Nichtlineare Verfahren 1/2

Support Vector Machine

In dieser Übung werden wir uns mit Support Vector Machines (SVM) beschäftigen. Dabei wird das Datenset für Iris-Blumen verwendet. Das Datenset besteht aus 150 Datenpunkten. Jeweils 50 dieser Datenpunkte gehören zu einer Klasse von Iris-Blumen. Klasse 0 ist Iris Setosa, Klasse 1 ist Iris Versicolor und Klasse 2 ist Iris Virginica. Jeder Datenpunkt hat 4 Features, 0 ist Sepal Length, 1 ist Sepal Width, 2 ist Petal Length und 3 ist Petal Width. In dieser Übung werden wir uns nur mit den letzten beiden Features beschäftigen. In Abbildung 1 werden diese beiden Features und die drei Klassen abgebildet. Blaue Punkte sind Setosa, gelbe Punkte Versicolor und braune Punkte Virginica.

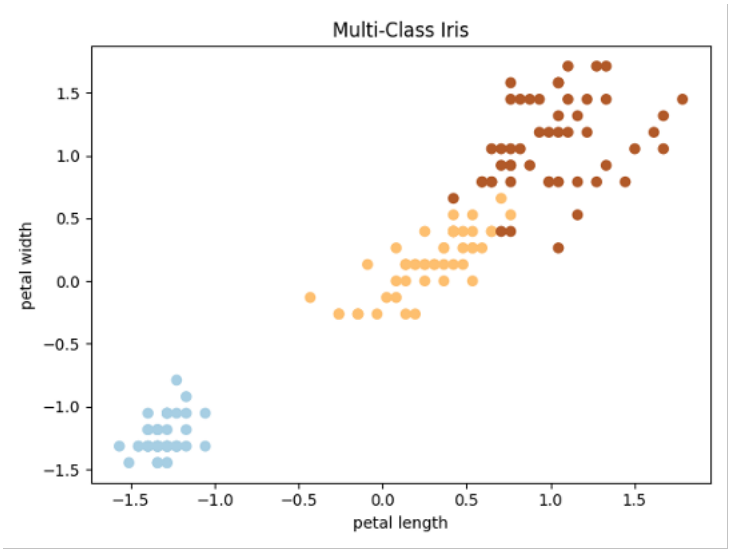

Abbildung 1

Teil 1

Für diesen Teil bearbeiten Sie *run_part_1.py*
In dieser Aufgabe geht es darum den Parameter *C* der SVM zu untersuchen. Für diese Aufgabe werden alle Datenpunkte der Klasse Setosa (0) ignoriert.
Wenn Sie diese Datei ausführen, werden Ihnen zwei Bilder und zwei Werte ausgegeben. Das erste Bild gibt die Datenpunkte an, mit denen wir arbeiten, während das zweite Bild die SVM zeigt. Die zwei Werte geben die Genauigkeit der trainierten SVM für das Training-/Test-Set an.

Aufgabe 1

Schauen Sie sich das zweite Bild genau an. Wofür steht die mittlere Linie? Wofür stehen die beiden gestrichelten Linien? Wofür stehen die eingekreisten Datenpunkte?
Geben Sie Ihre Antwort in *optional_1()* als String zurück.

Aufgabe 2

Ändern Sie nun den Parameter *C* der SVM (Zeile 33) und betrachten Sie wie sich die SVM verändert. Versuchen Sie die Werte [0.1, 1, 100].
Für welchen dieser Werte für *C* ist die Trainingsgenauigkeit am höchsten? Geben Sie Ihre Antwort als float in *question_1()* zurück.
Für welchen dieser Werte für *C* ist die Testgenauigkeit am höchsten? Geben Sie Ihre Antwort als float in *question_2()* zurück.

Teil 2

Für diesen Teil bearbeiten Sie *run_part_2.py*
In dieser Aufgabe werden wir den Kernel-Trick anwenden, um eine bessere Vorhersage zu bekommen. Betrachtet werden alle Datenpunkte, jedoch nur 2 Klassen. Zum einen die Klasse Iris Versicolor (1), zum anderen werden die Klassen Iris Setosa und Iris Virginica auf eine Klasse „nicht Iris Versicolor“ (0) abgebildet.
Wie in Teil 1 bekommen Sie wieder zwei Bilder und zwei Werte. Schauen Sie sich das erste Bild an, um die Ausgangslage des Problems zu verstehen.

Aufgabe 1

Wenn Sie die Bilder betrachten, welches Problem sehen Sie für die Klassifizierung von Iris Versicolor und nicht Iris Versicolor durch ein lineares Modell?
Geben Sie Ihre Antwort in *optional_2()* als String zurück.

Aufgabe 2

Um eine bessere Klassifizierung zu erhalten kann der Kernel-Trick angewandt werden, um weitere Features zur Klassifizierung zu simulieren.
Ändern Sie dazu den Kernel der SVM (Zeile 52) zu „poly“ oder „rbf“.
Für welchen Kerneltyp ist die Trainingsgenauigkeit am höchsten? Geben Sie Ihre Antwort als String in *question_3()* zurück.
Für welchen Kernel-Typ ist die Testgenauigkeit am höchsten? Geben Sie Ihre Antwort als String in *question_4()* zurück.

Teil 3

Für diesen Teil bearbeiten Sie *run_part_3.py*
In dieser Aufgabe werden alle drei Klassen verwendet und das Ziel ist eine Klassifizierung mittels SVM für alle drei Klassen. Da SVMs nur zwischen zwei Klassen unterscheiden können benötigen wir eine Strategie, um SVMs auf drei Klassen anzuwenden. Eine solche Strategie ist „One-versus-

Rest“ (OvR), bei der eine SVM für jede Klasse erzeugt wird, die vorhersagt, ob ein Datenpunkt zu dieser Klasse gehört oder nicht. In Teil 2 wurde eine solche SVM für die Klasse Iris Versicolor erzeugt. Ein Datenpunkt kann dann klassifiziert werden, indem jede SVM vorhersagt, ob der Datenpunkt zu der Klasse gehört.

Aufgabe 1

Da wir drei SVMs benötigen (für jede Klasse eine), benötigen wir auch drei verschiedene *target*-Vektoren, um diese SVMs zu trainieren.
Implementieren Sie die Funktionen *get_setosa_classifier_target_vector()*, *get_versicolor_classifier_target_vector()* und *get_virginica_classifier_target_vector()*. Die Funktionen geben jeweils einen *target*-Vektor zurück mit der eine SVM trainiert werden kann, um genau eine Klasse zu identifizieren.
Implementieren Sie die Funktion *train_svm_list()* in der Sie alle drei SVMs mit den zuvor erzeugten *target*-Vektoren trainieren.

Aufgabe 2

Schauen Sie sich die Ausgabe an. Jede Zeile enthält drei Werte, die angeben, ob dieser Datenpunkt zu Klasse 0, 1 oder 2 gehört.
Welches Problem sehen Sie hier? Gibt es Datenpunkte, die keiner Klasse oder mehreren Klassen zugeteilt werden? Wie könnte man dieses Problem lösen?
Geben Sie Ihre Antwort in *optional_3()* als String zurück.