Mini Use-Case 12: Support Vector Machine (SVM) für das klassifizieren von Bildern

In dem vorherigen Use-Case 11 (Klassifizierung mittels Logistischer Regression) wurde bereits ein Maschine Learning Modell verwendet, um Werkzeuge in einem Arbeitsbereich zu erkennen. Ein Kamerasystem wurde aufgesetzt, welches mittels logistischer Regression den Kamerastream Frame-für-Frame klassifiziert. Wie schon in dem Use-Case 11 angesprochen ist die Logistische Regression nicht das einzige Modell, mit dem eine solche Klassifizierung durchgeführt werden kann. Ein weiteres bekanntes Modell für einen solchen Task ist die sogenannte Support Vector Machine. Wie diese Funktioniert und implementiert wird werden wir in diesem Use-Case anschauen. Erneut werden wir einen Kamerastream klassifizieren – diesmal allerdings mit einer Support Vector Machine auch SVM genannt.

Die Aufbereitung der Daten (Bilder) ist ident zu der Aufbereitung im Use-Case 11. In Abbildung 1 ist diese erneut abgebildet. Hier wird ein RGB-Bild in den Notwendigen Vektor umgewandelt, welcher anschließend als Input für die SVM herangezogen wird. Mehr Informationen zu dieser Bildverarbeitung sind im Use-Case 11 enthalten.

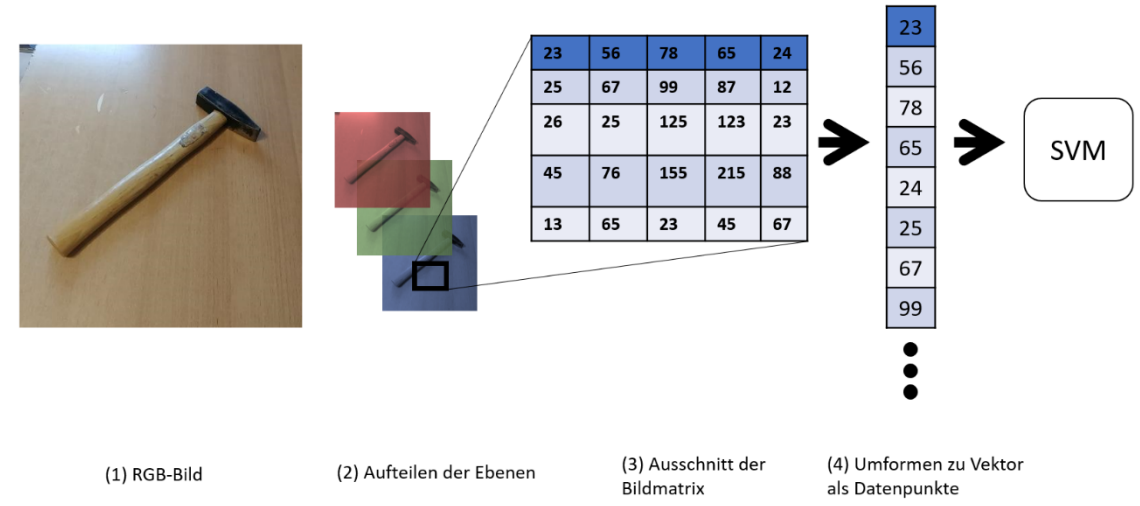

Abbildung 1: Aufbereiten der Daten (1) RGB-Bild welches in einen Spaltenvektor umgewandelt werden soll, (2) Aufteilung der Farbebenen in Rot, Grün und Blau, (3) Visualisierung eines Teiles der Bildmatrix, (4) Darstellung des Spaltenvektors im Vergleich zur Bildmatrix und Input für die Support Vector Machine)

Das Modell was für die Lösung dieser Klassifizierung verwendet wird heißt Support Vector Machine (SVM). Diese ist auch in den Videos SVM Teil 1 und SVM Teil 2 erklärt. Eine SVM versucht eine Hyperebene, also ein „Trennlinie“, zu berechnen, die Abhängig vom Datensatz ist, um verschiedene Klassen voneinander zu trennen. Woher kommt aber diese Hyperebene und was sagt sie aus? Zur Visualisierung einer SVM werden die Datenpunkte der beiden Klassen (Hammer bzw. Arbeitsplatz

ohne Hammer) in einem Diagramm – stark vereinfacht als Punkte – eingetragen wie in Abbildung 2 zu sehen ist. Die optimale Ebene wird anschließend durch das Maximieren des Abstandes der „äußersten Punkte der Klassen“ (Support Vektoren, daher auch der Name) bestimmt. In Abbildung 2 sind die Support Vectoren beschriftet. Umso größer der Abstand dieser Support Vektoren umso bessere Ergebnisse lassen sich erzielen. Dieser Abstand wird auch Margin genannt und darin liegt auch die Hyperebene.

Wie kann nun klassifiziert werden? Hierzu wird die folgende Gleichung (1) aufgestellt. Ein Normalvektor ω welcher Senkrecht zu der Hyperebene verläuft, beschreibt eine Gerade durch den Koordinatenursprung. Der Abstand zwischen Ursprung und dem Schnittpunk mit der Ebene nennt man Bias b. Mit diesen beiden Variablen lässt sich die folgende Gleichung aufstellen.

$$f(x) = sgn(\langle \omega, \mathrm{x} \rangle + b) \tag{1}$$

Diese Gleichung (1) sagt aus, dass alle Werte, die genau auf der Ebene liegen, 0 ergeben müssen. Ebenso kann jedem (neuen) zugehörigen Punkt x ein Wert y zugewiesen werden. Dieser entspricht für den Datensatz ±1 und kann so für die Klassifizierung eingesetzt werden. Diese mathematische Aufstellung ist in Abbildung 2 dargestellt. Bekommt man nun ein neues Bild, so kann erneut ein Vektor erzeugt werden, welcher in dem Diagramm eingesetzt wird und mittels der beschriebenen Formel klassifiziert werden [2].

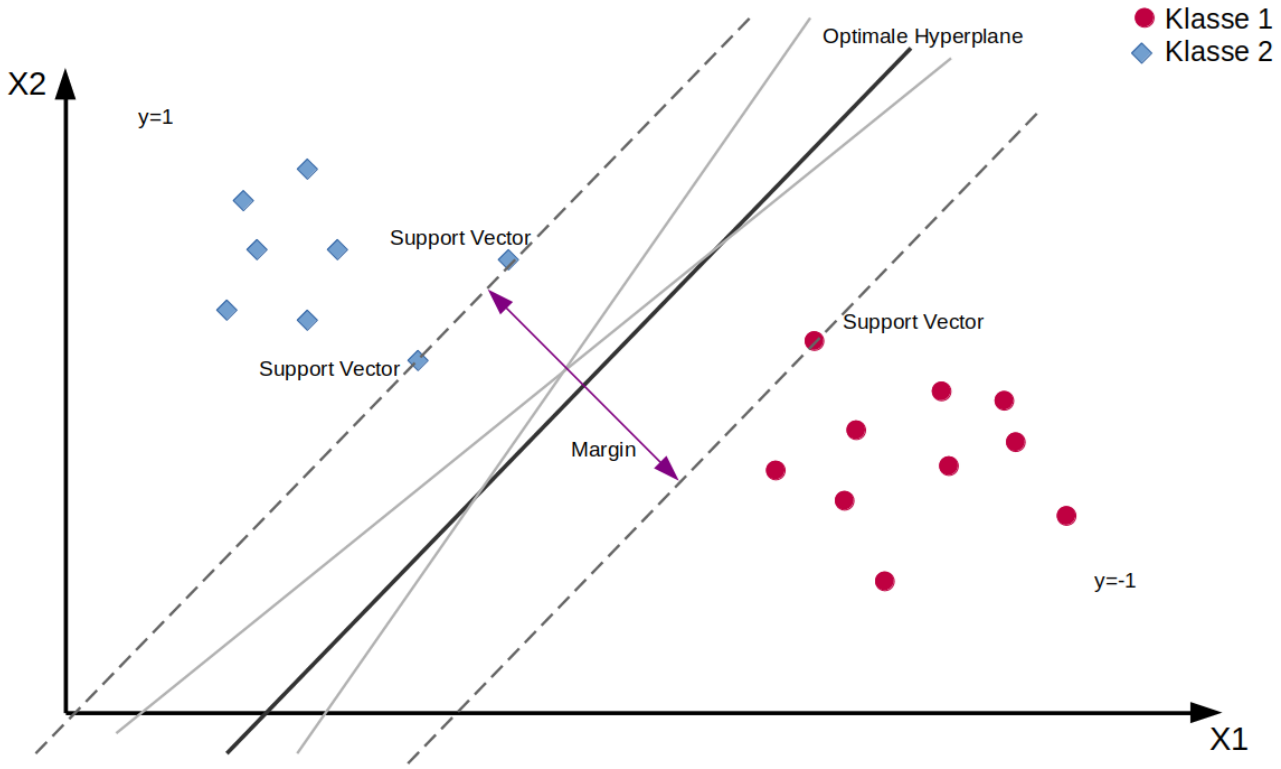

Abbildung 2: Darstellung einer SVM. Die beiden Klassen sind über die roten und blauen Datenpunkte dargestellt. Die optimale Hyperplane (durchgängige Linie, schwarz) liegt genau zwischen den äußersten Datenpunkten der beiden Klassen und versucht den Abstand zu maximieren (max. Margin). Unbekannte neue Datenpunkte können dann anhand dieser Plane klassifiziert werden.

Wie hilft dies nun mit dem vorgestellten Use-Case? Wie bereits angesprochen, geht es auch in diesem Use-Case darum einen Hammer im Arbeitsraum zu erkennen (wie auch schon im vorherigen Use-Case 11). Diese Klassifizierung kann mittels der gerade angesprochenen SVM durchgeführt werden

und ist ein weiteres Modell, der im Falle einer solchen Implementation durchaus eingesetzt werden kann. Durch das Trainieren im Beispielnotebook versucht das Modell die ideale Trennlinie/Hyperebene der beiden Klassen zu finden. Ideal beschreibt hier eine Maximierung des Abstandes der beiden Support Vektoren (Margin) [1] [3].

Für die Implementierung bietet die sklearn-Library verschiedene Kernel-Arten an. Dieses Kernel bestimmen grobgesagt wie Datenpunkte gruppiert werden. Mehr Informationen dazu sind im AIAV-Video SVM Teil 2 enthalten.

In Abbildung 2 ist eine mögliche Trennung zweier Klassen als Gerade dargestellt. Andere Funktionsarten sind beispielsweise eine Sigmoidfunktion wie wir sie schon im letzten Use-Case kennen gelernt haben. In dem beiliegenden Notebook wurde ein lineares Kernel eingesetzt (ähnlich wie die Abbildung 2). Mehr Informationen dazu findet man hier [4]. Für unseren Use-Case war der Lineare-Kernel ausreichend. Die 954 Bilder des Datensatzes wurden in einen Trainings- und Test-Datensatz aufgeteilt. 70% wurden herangezogen, um das Model zu trainieren. Die restlichen 30% der Bilder wurde verwendet um zu Messen wie gut das Model trainiert. Bevor allerdings die SVM trainiert wird, werden die Dimensionen der Trainings und Testdaten verringert. Hierzu wird erneut die PCA eingesetzt. Wie diese genau funktioniert und generell mehr Informationen zu der PCA sind in dem AIAV-Video Principal Component Analysis sowie in dem Use-Case Andocken autonomer mobiler Roboter erklärt. Als Messwert wie gut das Modell abschneidet kann mittels der Trainings- und Test-Daten ein „Model Accuracy Score“ ermittelt werden. Dieser hat in unserem Fall 0.93 (von 1.0) erreicht. Wenn nun das Ergebnis nicht zufriedenstellend ist, kann mittels der Kernel-Einstellungen und dem Datensatz versucht werden diesen Wert zu verbessern. Ebenso ist ein ausschlaggebender Parameter der SVM der C-Wert. Dieser gibt die Größe des Errors an, falls nicht eine 100% Trennung der Klassen möglich ist. Für den Use-Case wurde der Default-Wert von 1 verwendet. Mehr Informationen zu diesem Parameter sind in dem Video SVM Teil 1 erklärt.

Wird nun ein (Live-)Video ausgewertet, kann nun jeder Frame (Bild-für-Bild) der SVM gezeigt werden. Diese wertet das Bild aus (klassifiziert es) wie es auch in der Theorie besprochen wurde. Das Ergebnis dieser Frame-für-Frame Berechnung gibt dann bekannt, ob das Bild sich um Klasse 1 oder Klasse 2 handelt. Wird das Bild der Klasse „Hammer“ zugeordnet, so sagt es aus, dass die KI einen Hammer im Arbeitsraum erkannt hat und dementsprechend ein Alarm oder Hinweis ausgegeben werden soll.

Ebenso wie in dem vorherigen Use-Case (logistische Regression) kann auch hier der Datensatz in der Vorbereitung in ein Graustufenbild umgewandelt werden. Die Klassifizierung selbst kann so theoretisch schneller durchgeführt werden. Praktisch gesehen gab es kaum Veränderungen da es sich um ein recht simples Modell handelt. Genauso in der Genauigkeit der Klassifizierung konnten keine Verbesserungen festgestellt werden. Weitere Informationen zur Verbesserung wie Image Augmentation und einem Umstieg auf noch komplexere Modelle – Stichwort CNN – sind im Use-Case 11 zu finden. Ebenso sind die im vorherigen Use-Case angesprochenen Ängste einer „Überwachung“ zu beachten.

[1] Aberham J., Kuruc F. (2019) Support Vector Machine. In: Kersting K., Lampert C., Rothkopf C. (eds) Wie Maschinen lernen. Springer, Wiesbaden. https://doi-1org-10003429b00e9.han.technikum-wien.at/10.1007/978-3-658-26763-6_13

[2] Rolshoven, J. (2015). Textklassifikation mit Support Vector Machines. [Online]. Available: https://spinfo.phil-fak.uni-koeln.de/sites/spinfo/geduldia/SVM_Klassifikation.pdf. [Zugriff 16.11.2021]

[3] Nugroho, A. S., Witarto, A. B., & Handoko, D. (2003). Support vector machine. Proceeding Indones. Sci. Meeiting Cent. Japan. http://www.asnugroho.net/papers/ikcsvm.pdf

[4] scikit-learn developers, „scikit-learn.org“, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Zugriff am 5 Oktober 2021].