Mini Use-Case 13: k-Nearest Neigbors für das klassifizieren von Bildern

Wie auch schon in den beiden vorherigen Use-Cases 11 und 12, wird auch in diesem Use-Case ein Kamerasystem eingesetzt, um Werkzeug auf einem Tisch zu erkennen. Hierzu wird ein neues Modell vorgestellt, welcher bei der Implementierung erneut die gleiche Datenaufbereitung aufweist. Dies dient dazu, dass im Nachhinein die vier Klassifizierung-Use-Cases 11, 12, 13, 15 einfach verglichen werden können. Das Modell, der diesmal eingesetzt wird, heißt k-Neares Neigbor und wird nach Abbildung 1, wo eine Erinnerung an die Datenverarbeitung abgebildet ist, genauer erklärt. Das Modell benötigt wieder einen Vektor. Diesen Vektor erhält man, indem man die einzelnen Bildebenen aufteilt und von einer Matrix-Form in einen Vektor-Form bringt. Der eben angesprochene Vektor besteht aus sehr vielen Einträgen – genauer gesagt aus genau so vielen Einträgen wie die Bilder Pixel haben. Um diese Anzahl zu minimieren, wird eine PCA angewendet um die Dimensionen zu reduzieren. Wie eine PCA genau funktioniert und wie sie angewendet wird, wird in dem Video PCA und in dem Use-Case „Andocken autonomer mobiler Roboter“ genau erklärt. Der resultierende Vektor kann nun als Input für das Model verwendet werden. Genauere Informationen zu Abbildung 1 sind im Use-Case 11 Klassifizierung mittels logistischer Regression beschrieben.

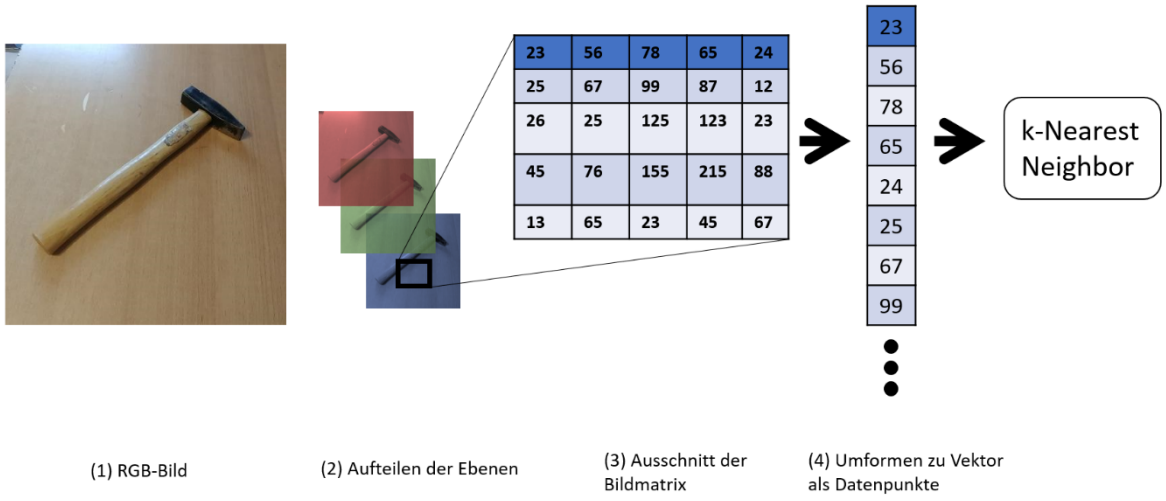

Abbildung 1: Aufbereiten der Daten ((1) RGB-Bild welches in einen Spaltenvektor umgewandelt werden soll, (2) Aufteilung der Farbebenen in Rot, Grün und Blau, (3) Visualisierung eines Teiles der Bildmatrix, (4) Darstellung des Spaltenvektors im Vergleich zur Bildmatrix und Input für das kNN Modells.

Wie bereits angesprochen wird in diesem Use-Case die Klassifizierung mittels k-Nearest Neigbors gelöst. Diese Methode ist eine Art des überwachten Lernens. Dieses Modell lernt über einen vorgegebenen Datensatz zu welcher Klasse welche Features gehören. Anschließend wird bei einem neuen Beispiel entschieden (klassifiziert) zu welcher bekannten Klasse dieser gehört. Dies geschieht über einen Parameter, den man ***k*** nennt. Daher kommt auch die Namensgebung k-Nearest Neigbor. Achtung im deutschsprachigen Raum kann *k*-NN leicht mit KNN verwechselt werden welches allerdings für Künstliches neuronales Netz steht. Dieser Parameter ***k*** entscheidet wie viele

benachbarte Datenpunkte analysiert werden sollen, um eine Entscheidung zu treffen zu welcher Klasse, den der neue Datenpunkt gehört.

Wie sieht das nun für die Arbeitsplatzüberwachung aus? Der Datensatz besteht aus Bildern von dem Arbeitsplatz mit Werkzeug und ohne Werkzeug. Per Hand wird jedes Bild in einen Ordner abgelegt, welcher „Hammer“ oder „Workspace“ heißt. Anschließend wird jedem Bild ein Label hinzugefügt, das dem Ordnernamen entspricht. Das bedeutet, dass jedes Bild, auf dem ein Hammer abgebildet ist, ein Label, also eine Art „Beschriftung“ bekommt, was dem Algorithmus sagt, dass auf dem vorliegenden Bild ein Hammer erkennbar ist. Das gleiche wird auch für die Bilder im Datensatz des Arbeitsbereiches (Workspace) durchgeführt. Zur Visualisierung werden die Datenpunkte in dem Diagramm – wie in Abbildung 2 sehr vereinfacht mit zwei Dimensionen nach der PCA dargestellt – eingetragen. Die beiden Farben Rot und Blau spiegeln die Klassen wider. Hier ist beispielsweise die Klasse 1 (rot) ein Bild, bei dem ein Hammer im Arbeitsbereich ist und bei Klasse 2 (blau) ist der Arbeitsbereich frei von Werkzeugen. Wird nun ein neuer Datenpunkt evaluiert (Prediction eines Bildes vom Videostream, grüner Datenpunkt) muss entschieden werden zu welcher Klasse dieser gehört. In der Abbildung 2 entspricht ***k*** gleich 4 und somit werden die 4 nahestehen Datenpunkte evaluiert. Da 3-mal die Klasse 1 vorkommt und nur 1-mal die Klasse 2, wird daraus geschlossen, dass der neue Datenpunkt ebenfalls zu Klasse 1 gehört.

Der Parameter k wird schon beim Lernen des Modells angegeben und ist auch eine der einfachsten Variablen zum Verfeinern des Modells. Ein kleiner Wert für k ist sehr „streng“ und zieht vielleicht zu wenige Punkte in Betracht. Das kann bei sehr dicht aneinander liegenden Datenpunkten ein Problem sein. Wählt man ein größerer Wert für k, so kann eine Art „Mittelwert“ aus den Umliegenden Datenpunkten genommen werden. Wichtig ist es hier den Mittelweg zu finden und ein k zu wählen, welches auf den eigenen Datensatz passt. Im Beispiel in Abbildung 2 ist k mit 4 gewählt und somit ist neue Datenpunkt, wie schon besprochen, der Klasse 1 zuzuordnen. Was passiert, allerdings wenn k den Wert 2 hat? Hier wird dann vermutlich ein Punkt der Klasse 1 und ein Punkt der Klasse 2 berücksichtigt. Nun ist es nicht mehr eindeutig zu welcher Klasse der neue Datenpunkt gehört. [1]

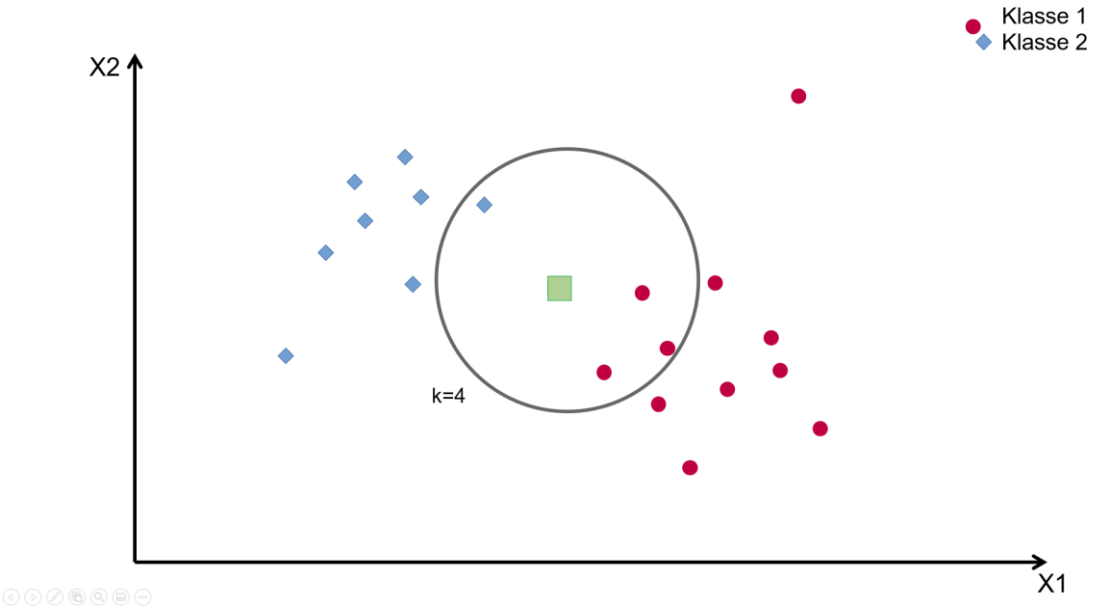

Abbildung 2: Darstellung des k-NN mit k=4, 2 Klassen (rot, blau) und einem neuen Datenpunkt (grün)

Das beiliegende Code-Beispiel ist auf genau dieser Theorie des k-NN aufgebaut und für eine Hammer

Klassifizierung mittels der sklearn-Library implementiert [2]. Über eine Kamera wird mittels k-NN der Videostream klassifiziert ob sich ein Hammer in dem Arbeitsbereich befindet oder nicht. Um einen eigenen Datensatz zu trainieren, können einfach neue Bilder mit neuen Klassen in der vorgegebenen Ordnerstruktur abgelegt werden. Diese Bilder werden in dem Notebook erneut aufgeteilt in Trainings- und Test-Datensatz. Das bedeutet, dass in unserem Fall 75% aller 954 Bilder verwendet, werden um das Model zu Trainieren. Mit den restlichen 25% wird das Model getestet, um zu schauen, wie gut das Modell funktioniert. Damit lässt sich nicht nur das Modell trainieren, sondern auch erneut ein „Accuracy Score“ berechnen. Dieser hat bei unserem Modell 0.92 von 1.0 ergeben. Ebenso werden die Daten, bevor sie zum Trainieren verwendet, werden mit einer PCA analysiert. Diese ist wie auch in dem Use-Case 11 implementiert. Mehr Informationen dazu sind hier zu finden sowie in dem Use-Case 11.

Dadurch, dass dieser Use-Case genauso aufgebaut und strukturiert ist wie die bereit angesprochenen Use-Cases Logistische Regression und SVM (11 und 12) kann eine Verbesserung des Ergebnisses auf dieselbe Art wie in dieses Use-Cases erzielt werden. Mehr Informationen dazu sind bei den jeweiligen Use-Cases zu finden.

[1] Neumann M. (2019) k-Nächste-Nachbarn. In: Kersting K., Lampert C., Rothkopf C. (eds) Wie Maschinen lernen. Springer, Wiesbaden. https://doi-1org-10003429b00f2.han.technikum-wien.at/10.1007/978-3-658-26763-6_10

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