Wie wir Grenzen finden - k-Nearest Neighbour

Storyboard

In den Micro Usecases TODO (Moritz CNN) und TODO (Moritz SVM) haben wir uns bereits mit Methoden der Klassifizierung beschäftigt (siehe AIAV Video Classifier vs. Detector). Die dort gezeigten Modelle erkennen ob sich ein Hammer im Sichtfeld einer Kamera befindet. Dafür wurden die Modelle zunächst anhand eines beschrifteten Datensatzes (siehe Abbildung 1), bestehend aus Bildern von Hämmern und Bildern ohne Hämmer, trainiert. Da der Trainingsdatensatz beschriftet ist, weist jedes Bild darin eine Beschriftung auf. Diese gibt an, ob sich im jeweiligen Bild ein Hammer befindet oder nicht. Das ist die sogenannte Klasse, welche wir später erkennen wollen. Nach Abschluss der Trainingsphase werden neue, unbekannte Bilder in das trainierte Modell gegeben, welches entscheidet ob im Bild ein Hammer zu sehen ist, oder nicht.

Das Modell ist nach dem Training in der Lage, Bilder mit Hämmern von Bildern ohne Hämmer zu unterscheiden. Das heißt also, dass diese Modelle in der Trainingsphase eine Grenze zwischen den Klassen in den Trainingsdaten finden. In unseren Beispielen ist diese Grenze zwischen Bildern, welche einen Hammer enthalten und Bildern ohne Hammer. Aber wie sehen solche Grenzen aus und wie werden sie gefunden?

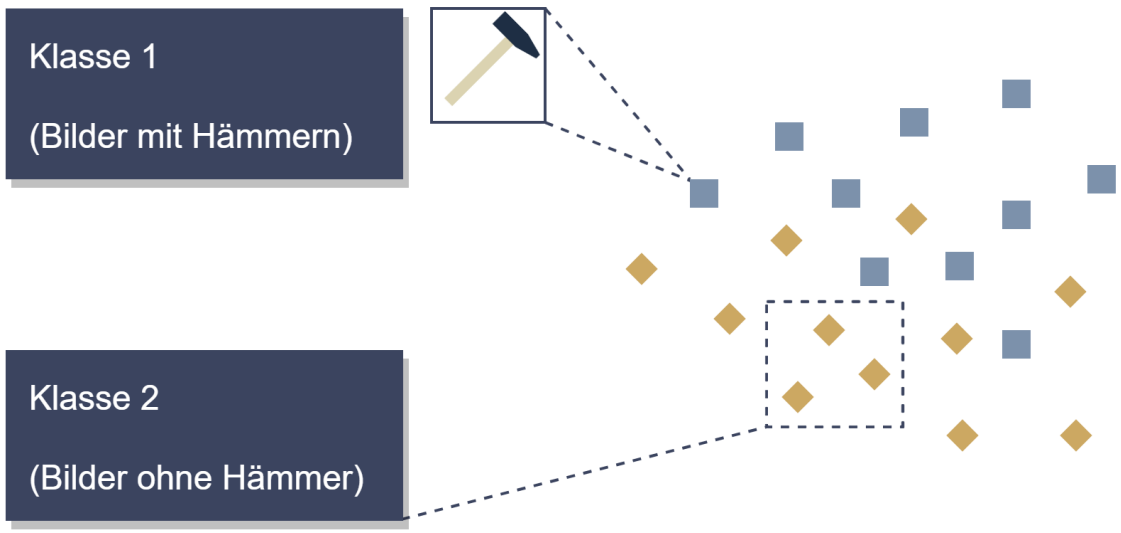

Abbildung 1: Dieser beilspielhafte Trainingsdatensatz besitzt mehrere Einträge auf einer Ebene, welche einer von zwei Klassen (blaue Quadrate oder goldene Raute) angehören. Wir stellen uns die Frage, wie man in diesem Datensatz eine Grenze zwischen den Klassen finden kann, um neue, unbekannte Punkte einer der Klassen zuzuordnen.

Ein Verfahren, welches dieses Problem sehr einfach und zuverlässig löst, ist k-Nearest Neighbour (kNN). kNNs ermitteln die Grenzen zwischen Klassen geometrisch; die Grenze wird anhand des Abstandes zwischen benachbarten Dateneinträgen gefunden. Dafür werden beschriftete Trainingsdaten zunächst gespeichert. Soll nun ein Punkt klassifiziert werden, werden alle bekannten Datenpunkte in einem festlegbaren Radius k rund um den Punkt gezählt. Diese gezählten Einträge sind dann dei Grundlage für eine Mehrheitsentscheidung - die Klasse mit den am öftesten innerhalb des Radius vorkommenden Datenpunkten gewinnt. So Entsteht eine klare Trennung zwischen den verschiedenen Klassen basierend auf beschrifteten Trainingsdaten. Das k, also der maximale Abstand

in dem Nachbarn berücksichtigt werden, ist dabei bei den meisten Implementierungen wählbar und beeinflusst das Verhalten der Klassifizierung stark.

Abbildung 2 zeigt das Konzept hinter kNNs für den Datensatz in Abbildung 1 mit zwei verschiedenen k Werten. Dabei versuchen wir den zu klassifizierenden Eintrag (grüner Punkt) einer der beiden Klassen zuzuordnen. Die verschiedenen Werte für k stellen dabei den Radius um den Eintrag dar, in dem benachbarte Datenpunkte gezählt werden. Je größer das k ist, desto mehr Nachbarn werden also gezählt. Dabei sieht man die Auswirkung, welche die Wahl von k auf den Ausgang der Klassifizierung hat. Beim kleinen k wird der Eintrag Klasse 2 zugeordnet, während große Werte für k in einer Zuordnung zu Klasse 1 resultieren.

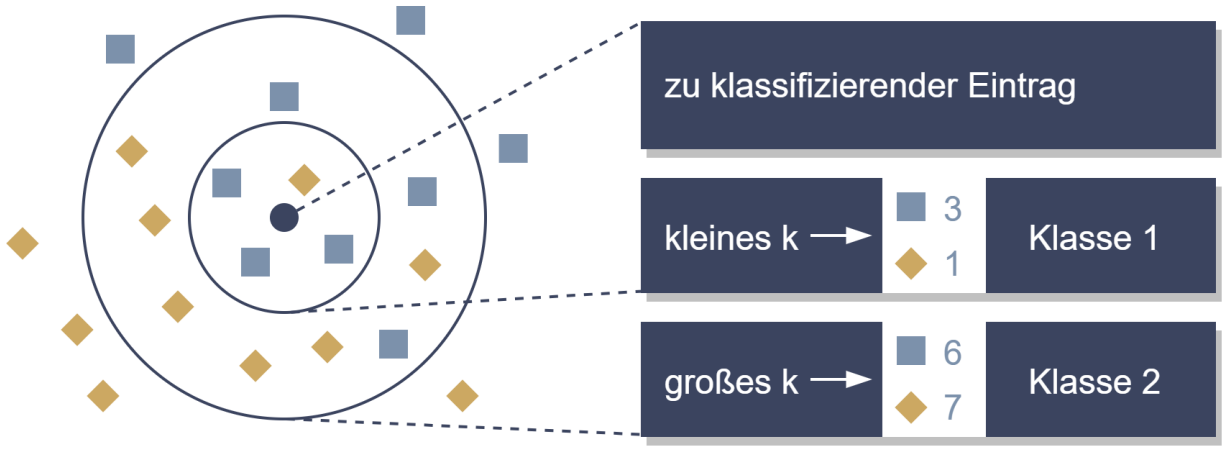

Abbildung 2: Der k-Nearest Neighbour Algorithmus erlaubt es uns neue Einträge Klassen zuzuordnen indem alle bekannten Datenpunkte in einem Bestimmten Radius gezählt werden. Die Klasse, die am öftesten innerhalb des Radius vorkommt, gewinnt. Dabei stellt k den Radius, in dem gezählt wird, dar. Für verschiedene Werte von k kann die Klassifizierung verschiedene Werte liefern.

Praktische Implementierung

Für die Praktische Implementierung dieses Usecases wollen wir Vorder- und Hintergrund eines Bildes trennen. Dies passiert mithilfe des kNN Algorithmus, welcher es uns wie oben beschrieben erlaubt die Grenze zwischen mehreren Klassen zu finden. Das nutzen wir, indem wir Vorder- und Hintergrund je als eigene Klasse definieren. Die Idee ist es, dann mittels kNN im Bild die Grenzen zwischen den beiden Klassen zu ermitteln, um die verschiedenen Bildbereiche dem Vorder- oder Hintergrund zuzuordnen.

Diese Implementierung benötigt ein Bild vom Hintergrund, welches zunächst aufgenommen und gespeichert wird. Dieses Bild wird mit den zu analysierenden Bildern verglichen; die Unterschiede zwischen den Bildern sind dabei der Vordergrund und die Gemeinsamkeiten der Hintergrund. Verglichen wird dabei mittels des Structural Similarity Indexes, welcher für jeden Bildpunkt einen Wert zurückgibt. Dieser Wert gibt an, wie sehr sich die beiden Bilder, also das Bild vom Hintergrund und das zu analysierende Bild, an der jeweiligen Position ähneln. Das Resultat ist ein "Wärmebild" welches angibt, wie sicher es ist, dass die Bilder unterschiedlich sind. Dieses Wärmebild wird anschließend auf eine niedrigere Auflösung skaliert, um den erforderlichen Rechenaufwand gering zu halten. Dannach setzen wir alle Werte unter einem Schwellenwert auf 0 und alle Werte über dem Schwellenwert auf 1.

Grundsätzlich sind mit diesen Verarbeitungsschritten bereits alle Bildpunkte entweder dem Vorder- oder Hintergrund zugeordnet, da die resultierende Karte angibt, welcher Klasse der jeweilige Bildpunkt angehört. Einträge mit dem Wert 1 gehören zum Hintergrund, während Einträge mit dem Wert 0 zum Vordergrund gehören. Dabei weist die Karte aber oft Artefakte und unschöne Kanten um das zu erkennende Objekt auf. Diese entstehen durch sich änderende Lichtverhältnisse sowie die Verarbeitung des Bildes direkt in der Kamera. Damit wir nun schöne Grenzen zwischen Vorder- und Hintergrund erhalten und Artefakte beseitigen, verwenden wir kNN Klassifizierung. Dafür wird ein kNN Klassifikator aus der Scikit Learn Bibliothek anhand der Karte trainiert. Wie oben beschrieben, werden dann die Grenzen zwischen den beiden Klassen anhand der Nachbarn jedes Eintrags berechnet. Je höher die Anzahl der Nachbarn, desto gleichmäßiger sind die berechneten Grenzen. Bei zu hoher Anzahl an Nachbarn kann es aber vorkommen, dass Details im Umriss um ein zu erkennendes Objekt weggeglättet werden. Zusätzlich dazu steigt mit wachsender Anzahl der Nachbarn auch der benötigte Rechenaufwand erheblich.

Die Maske aus den von der kNN Klasse ermittelten Grenzen wird wieder auf die originale Größe der eingehenden Bilder hochskaliert und auf das eingehende Bild angewendet. Dabei werden nur als Vordergrund gekennzeichnete Teile des Bildes übernommen und als Hintergrund gekennzeichnete Teile verworfen. Damit ergibt sich ein Bild, welches nur den Vordergrund enthält. Der gesamte Ablauf der Bildverarbeitungspipeline ist in Abbildung 3 mit Visualisierungen von jedem Verarbeitungsschritt dargestellt.

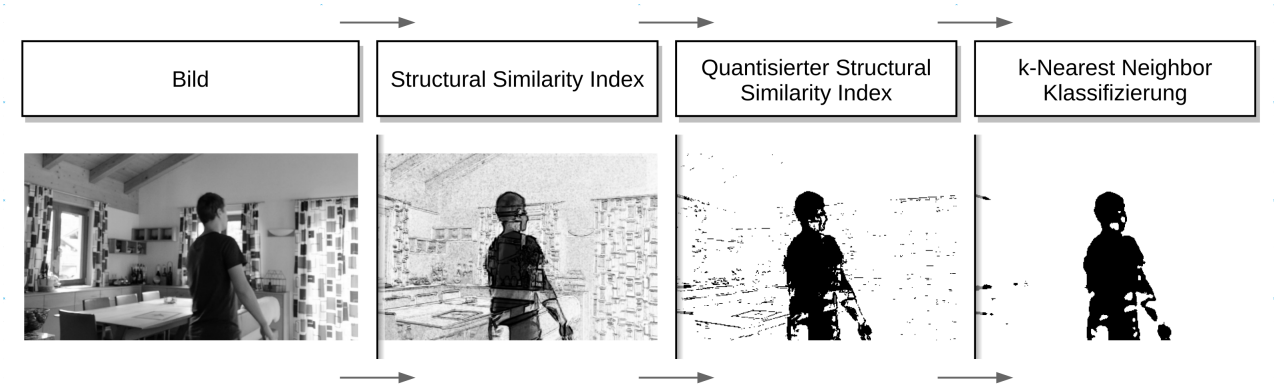

Abbildung 3: Um Vorder- und Hintergrund zu trennen wird das zu analysierende Bild zunächst mit einem Bild des Hintergrunds verglichen. Der aus dem Vergleich resultierende Structural Similarity Index (SSIM) wird anhand eines Schwellenwertes quantisiert. Mittels kNN werden dann Artefakte im SSIM entfernt, welche durch sich ändernde Lichtverhältnisse und Bildverarbeitung in der Kamera verursacht wurden.

@Willi: Die Bilder hier sind nur als Platzhalter drinnen. Da werde ich im C Turm noch welche mit industrienahem Setting aufnehmen

Fazit

k-Nearest Neighor erlaubt es uns Klassifizierung anhand eines relativ kleinen Datensatzes zu betreiben, um die Grenzen zwischen Klassen zu ermitteln. Dadurch dass das Konzept hinter kNN relativ einfach ist, ist nur ein geringer Rechenaufwad für Training und Klassifizierung notwendig. Ein weiterer Vorteil von kNN ist der niedrige Notwendige Implementierungsaufwand, da diese Art von Klassifikation bereits in Form von gängigen Bibliotheken, wie Scikit Learn oder nativ in der Programmiersprache R verfürbar ist.