1 Anwendung: Handschriftenerkennung

Die Signatur besitzt zahlreiche praktische Anwendungsfelder. Eines davon ist das Erkennen von Mustern, in unserem Fall von gezeichneten Zahlen. Ziel dieses Abschnitts wird es sein, ein Programm zu entwickeln, das als Eingabe einen Pfad bekommt und dann mit Hilfe der Signatur deuten kann, um welchen Buchstaben es sich handelt.
Etwas Ähnliches tat der Engländer Benjamin Graham im Jahr 2013. Er entwickelte ein Programm, das unter anderem die Signatur nutzt, um gezeichnete Chinesische Schriftzeichen zu erkennen ([?]). Dieses gewann die „12. Internationale Konferenz zur Dokument Analyse und Erkennung“*(ICDAR)* bei der es Schriftzeichen mit über 97%iger Genauigkeit erkannte.
Wir betrachten hier ein viel simpleres Beispiel, an dem man aber schon die wichtigen Ideen nachvollziehen kann. Wir wollen ein Programm entwickeln, dass die gezeichneten Zahlen von $0-9$ erkennen kann. Zur Umsetzung verwenden wir die Programmiersprache Python und die interne Bibliothek `pygame`.
Da wir natürlich nicht unendlich viele Signaturlevels berechnen können, beschränken wir uns hier nur auf die ersten vier.
Als Erstes erstellen wir ein $N \times N$ großes weißes Gitter. Sobald die Maus gedrückt wird, soll das Programm diejenigen Zellen schwarz färben, über denen sich der Mauszeiger befindet. Am Ende erkennt man dann das Bild eines Pfades X, welches der Form einer Zahl entsprechen soll. Der Pfad soll wieder auf $[0,1]$ parametrisiert sein, sodass X_0 das zuerst und X_1 das zuletzt markierte Feld ist.

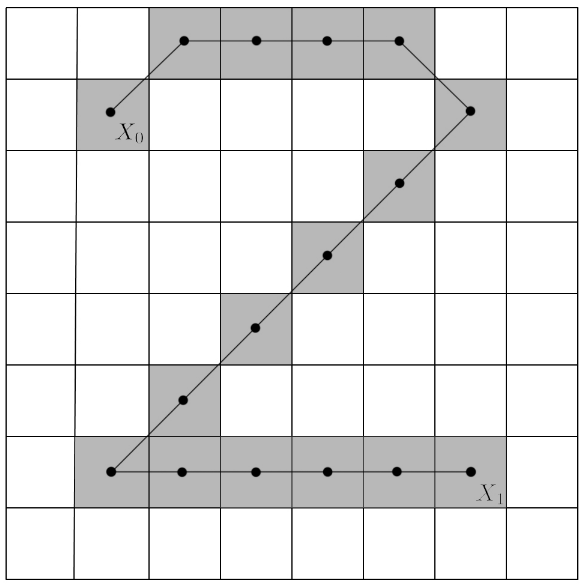


Abbildung 1: Beispiel für eine gezeichnete 2 mit $N = 8$

Das bedeutet, dass unserer Pfad aus stückweise affin linearen Pfaden besteht. Für einen affin linearen Pfad Y_t von $\binom{x_1}{y_1}$ nach $\binom{x_2}{y_2}$ ist die Signatur sehr einfach zu berechnen. Dazu machen wir uns zuerst Gedanken über das Volumen des Standartsimplex $\Delta^n_{a,b}$, (siehe [?]).
Wir wissen, dass das Volumen des Standarwürfels $\mu([a,b]^n) = (b-a)^n$ ist. Jetzt zerlegen wir den Würfel in $n!$ gleich große Simplexe durch

$$[a,b]^n = \dot{\bigcup_{\sigma \in \mathfrak{S}_n}} \left\{ (u_1, \ldots, u_n) \;\middle|\; a < u_{\sigma(1)} < u_{\sigma(2)} < \cdots < u_{\sigma(n)} < b \right\}.$$

Also ist $\mu(\Delta^n_{a,b}) = \frac{1}{n!}(b-a)^n$ und wir können die Signatur von Y nun ausrechnen.

$$\pi_n(\mathsf{S}(Y)) = \int\limits_{a<u_1<\dots<u_n<b} \dot{Y}^{\otimes n}\,\mathrm{d}u_1 \dots \mathrm{d}u_n = \frac{(b-a)^n}{n!}\dot{Y}^{\otimes n} \tag{1}$$

Algorithmus 1 Berechnung der Signatur von linearen Pfaden

Input: x_1, y_1, x_2, y_2
Output: Signatur des Pfades von $\left[\binom{x_1}{y_1}, \binom{x_2}{y_2}\right]$

for $n = 1, \dots, 4$ **do**
    for $i_1, \dots, i_n \in \{1,2\}^n$ **do**
        $S(X)^{i_1,\dots,i_n}_{\left(\binom{x_1}{y_1},\binom{x_2}{y_2}\right)} = \frac{1}{n!}\cdot(x_2-x_1)^{\text{Anzahl der 1er in }(i_1,\dots,i_n)}\cdot(y_2-y_1)^{\text{Anzahl der 2er in }(i_1,\dots,i_n)}$
    end for
end for

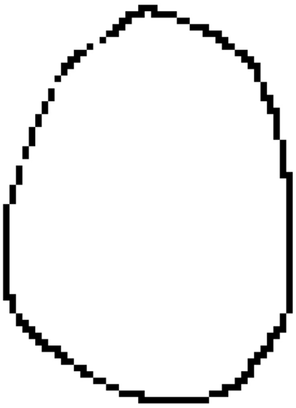
Um nun die Signatur von X zu berechnen, müssen wir in jedem Schritt, in dem wir ein Feld markieren, die Signatur des linearen Pfades vom aktuellen zum letzten Feld berechnen und dann das Tensorprodukt mit der bisherigen Signatur bilden. Nach Chen's Identität (**??**) ist das Ergebnis dann die Signatur des gesamten Pfades.

Algorithmus 2 Berechnung der Signatur von allgemeinen Pfaden

Signatur $\leftarrow \mathbb{1}$
$x\_alt \leftarrow y\_alt \leftarrow -1$
while Maustaste gedrückt **do**
    if Zelle nicht markiert **then**
        Zelle markieren
        Bestimme x_neu und y_neu
        if $x\_alt\ != -1$ **then**
            $Signatur \leftarrow Signatur \otimes berechnete\_signatur(x\_alt, y\_alt, x\_neu, y\_neu)$
        end if
        $x\_alt \leftarrow x\_neu$
        $y\_alt \leftarrow y\_neu$
    end if
end while

Der Vorteil dieses Verfahrens ist, dass egal wie lang der Pfad wird, wir uns immer nur eine Signatur mit konstanter Größe merken müssen. Nun, da wir also die Signatur von X bestimmt haben, können wir einfach einmal jede Zahl zwischen 0-9 zeichnen und uns die Signatur notieren. Wenn wir danach nochmal eine Figur zeichnen, müssen wir nur ihre Signatur bestimmen und abgleichen, welcher der vorher gespeicherten Signaturen dieser am ähnlichsten ist und schon wissen wir um welche Zahl es sich handelt. Wir minimieren hier einfach die Norm zu den vorher errechneten Werten, aber Graham benutzt dafür ein *Convolutional Neuronal Network (künstliche Intelligenz)*, um zu entscheiden, zu welchem Symbol die errechnete Signatur gehört.
Betrachten wir nun als Beispiel ein 70×70 großes Gitter. Dann liefert das Programm die folgenden Ergebnisse, wobei wir die (gerundete) Signatur hier, um Platz zu sparen, schreiben als

$$\mathsf{S}(X) = \left(\begin{pmatrix} S(X)^1 \\ S(X)^2 \end{pmatrix}, \begin{pmatrix} S(X)^{1,1} & S(X)^{1,2} \\ S(X)^{2,1} & S(X)^{2,2} \end{pmatrix}, \begin{pmatrix} S(X)^{1,1,1} & S(X)^{1,1,2} \\ S(X)^{1,2,1} & S(X)^{1,2,2} \\ S(X)^{2,1,1} & S(X)^{2,1,2} \\ S(X)^{2,2,1} & S(X)^{2,2,2} \end{pmatrix} \right).$$



(a) Gezeichnete Null

$$\mathsf{S}(\text{„0“}) = \left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0 & 2100\\2100 & 0.5\end{pmatrix}, \begin{pmatrix}0 & 8284\\-16568 & 63292\\8284 & -128683\\65391 & 0\end{pmatrix}, .. \right)$$

Abstände:

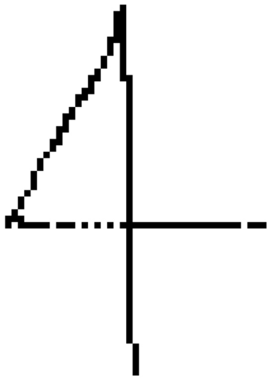
3949: [0], 7345: [1], 7011: [2], 7788: [3], 7079: [4], 8078: [5], 6134: [6], 8913: [7], 8140: [8], 7989: [9]

(b) Gezeichnete Zwei

$$\mathsf{S}(\text{„2“}) = \left(\begin{pmatrix}44\\-42\end{pmatrix}, \begin{pmatrix}968 & -730\\-1118 & 882\end{pmatrix}, \begin{pmatrix}14197 & -7840\\-16462 & 31292\\-16354 & -31903\\39419 & -12348\end{pmatrix}, .. \right)$$

Abstände:

3768: [0], 3104: [1], **1361: [2]**, 3228: [3], 3505: [4], 3475: [5], 3011: [6], 4222: [7], 3918: [8], 3484: [9]

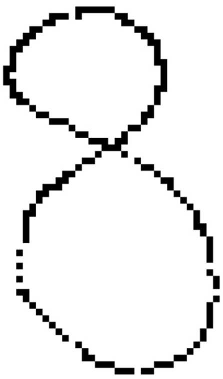


(c) Gezeichnete Vier

$$\mathsf{S}(\text{„4“}) = \left(\begin{pmatrix}20\\23\end{pmatrix}, \begin{pmatrix}200 & 172\\289 & 265\end{pmatrix}, \begin{pmatrix}1333 & -3455\\10341 & -4797\\-2286 & 13541\\-3452 & 2028\end{pmatrix}, .. \right)$$

Abstände:

3199: [0], 1198: [1], 2424: [2], 1853: [3], **275: [4]**, 1463: [5], 1238: [6], 2683: [7], 1550: [8], 1441: [9]



(d) Gezeichnete Acht

$$\mathsf{S}(\text{„8“}) = \left(\begin{pmatrix}1\\-1\end{pmatrix}, \begin{pmatrix}0.5 & 392\\-394 & 0.5\end{pmatrix}, \begin{pmatrix}0 & -1637\\3668 & -25243\\-2031 & 50094\\-24850 & 0\end{pmatrix}, .. \right)$$

Abstände:

4946: [0], 2423: [1], 3510: [2], 2522: [3], 2162: [4], 1906: [5], 2918: [6], 3941: [7], **879: [8]**, 1862: [9]

Abbildung 2: Erkennung von Zahlen in python