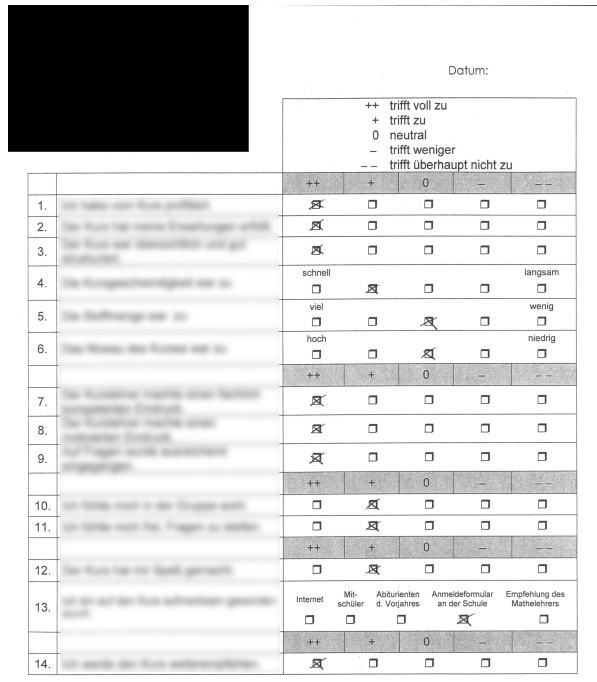


Einführung in Python

Abschlussprojekt Neuronale Netze als Evaluationshelfer

Dass Ihre Nachhilfekurse phänomenal sind, ist Ihnen natürlich klar. Es schadet jedoch nie, harte Fakten auf dem Tisch zu haben. Aus diesem Grunde entschließen Sie sich dazu, die Kurse ebenfalls von den Teilnehmern beurteilen zu lassen. Die anonymisierten gescannten Fragebögen haben ein einheitliches Format und sehen aus wie folgt:



Nun wollen Sie sicherstellen, dass keiner Ihrer hochbezahlten Mitarbeiter durch die ermüdende Arbeit vom stundenlangen Kreuze zählen demotiviert wird. Deswegen soll das Auszählen und Generieren der wertvollen Statistiken vollautomatisch passieren.

Hinweis: Informieren Sie sich über das Python-Modul `Pillow`.

Aufgabe 1 (Referenzbogen erstellen)

Drucker und Scanner arbeiten nie ganz zuverlässig! Mit unterschiedlichen Pixelwerten ist ebenso zu rechnen, wie mit leichten Verschiebungen und Drehungen.

Um ein Gefühl für diese Fehler zu bekommen, betrachten Sie ein paar der gescannten Bögen in der Sammlung `boegen.zip`.

- Wählen Sie einen der Bögen als Referenzbogen und legen Sie einen Datensatz an, der zu jeder Frage die genaue Position der zugehörigen ankreuzbaren Kästchen enthält.
- Wählen Sie mindestens vier markante Punkte, wie zum Beispiel die Ecken des äußeren Rahmens eines Antwortfeldes, und fügen Sie die genaue Position dieser “Referenzpunkte”, die “Referenzpositionen”, zu dem Datensatz hinzu.

Um die Kästchen in einem der anderen Bögen mithilfe des Referenzbogens zu finden, müssen wir wissen, wie der Bogen gegenüber dem Referenzbogen verzerrt ist. Diese Transformation modellieren wir als Abbildung $\phi^{A,b}: \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $x \mapsto Ax + b$, wobei $b \in \mathbb{R}^2$ und $A \in \mathbb{R}^{2 \times 2}$.

- Schreiben Sie eine Funktion, die in einem bestimmten Bereich eines Bildes nach einem Referenzpunkt sucht und ggf. die gefundene Position zurückgibt.

Beispielsweise können Sie eine Schablone wie in Abbildung 1 über das Bild legen und die Quadratsumme der pixelweisen Differenzen betrachten. Die Position, für die diese Differenz am kleinsten ist, zeigt dann wahrscheinlich das Muster auf der Schablone.



Abbildung 1: Vier Schablonen für vier verschiedene Ecktypen, vgl. `masks.zip`.

- Schreiben Sie eine Funktion, die mithilfe der Referenzpunkte \hat{p}^j auf dem Referenzbogen und den entsprechenden Positionen p^j , $j = 1, \dots, J$, auf einem anderen Bogen die Matrix A und den Vektor b so bestimmt, dass

$$\mathcal{E}(A, b) = \sum_{j=1}^J \|\phi^{A,b}(\hat{p}^j) - p^j\|_2^2$$

minimal wird. Verwenden Sie dazu mindestens $J = 4$ markante Positionen und betrachten Sie das Problem auf dem Vektorraum $\mathbb{R}^{2 \times 2} \times \mathbb{R}^2$.

(Hier ist $(A, b) \in \mathbb{R}^{2 \times 2} \times \mathbb{R}^2$ die Unbekannte).

Finden Sie dazu abhängig von (\hat{p}^j, p^j) , $j = 1, \dots, J$, eine Matrix $C \in \mathbb{R}^{2J \times 6}$ und einen Vektor $b \in \mathbb{R}^{2J}$, sodass Sie $\mathcal{E}(A, b)$ als $\|Cx^{A,b} - b\|_2^2$ schreiben können. Der Vektor $x^{A,b} \in \mathbb{R}^6$ enthält dabei die Einträge von A und b als Komponenten.

Mit der so erhaltenen Transformation $\phi^{A,b}$ können die Positionen der Kästchen auf dem Referenzbogen in Positionen auf dem anderen Bogen übertragen werden.

Aufgabe 2 (Kästchen finden)

Um die angekreuzte Antwort auf eine Frage herauszufinden, müssen Sie die zugehörigen Kästchen aus dem Bogen extrahieren. Schreiben Sie dazu Teilprogramme, die folgendes leisten.

- Für jede der Antwortmöglichkeiten von + + bis - - schneiden Sie das zugehörige Kästchen großzügig als eigenes Bild heraus.
- Für spätere Verarbeitung ist es wichtig sicherzustellen, dass diese Ausschnitte zu den vorhandenen Daten passen. Betrachten Sie dazu Beispiele aus der Sammlung [crosses.zip](#).
- Skalieren Sie das ausgeschnittene Bild des Kästchens auf eine fixe Größe von 40x40 Pixel.



Abbildung 2: Leeres und angekreuztes Kästchen (zur Verdeutlichung mit Rahmen).

Doch wie finden wir nun heraus, was es bedeutet *angekreuzt* zu sein? Die Kreuze sind kreuz und quer und mit sehr verschiedenen Stiften gemalt...

Aufgabe 3 (Neuronales Netzwerk)

Angekreuzt sein oder nicht angekreuzt sein, das ist hier die Frage. Wir stehen also vor einem *binärem Klassifizierungsproblem*.

Für solche (und viele andere) sind in den letzten Jahren die sogenannten *Deep Neural Networks* immer beliebter geworden. Dieser Algorithmus schafft es, gefüttert mit bereits klassifizierten Daten, durch automatisches Lernen, d. h. Justieren der Gewichte und Bias-Vektoren (s. u.), ein Modell zu finden, welches die Klassifizierung vornimmt.

Ein neuronales Netzwerk ist ein gerichteter Graph, dessen Knoten (*Neuronen*) durch gewichtete Kanten miteinander verbunden sind. Die Neuronen sind dabei in mehrere Schichten aufgeteilt, wobei es immer nur Verbindungen zwischen zwei aufeinanderfolgenden Schichten gibt, vgl. Abbildung 3.

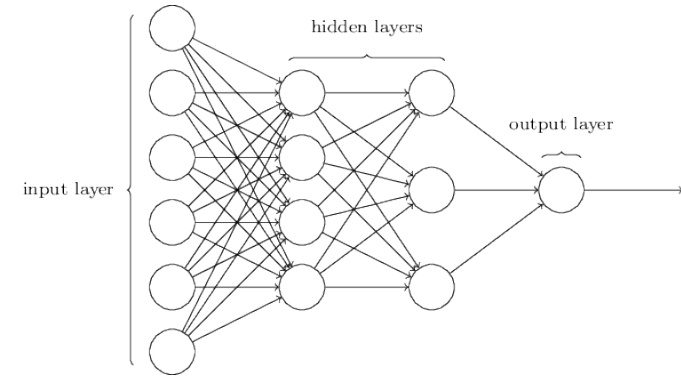


Abbildung 3: Ein neuronales Netz mit vier Schichten und sechs Eingabe- sowie einem Ausgabeneuron. (Quelle [1])

Jedes Neuron kann als Funktion aufgefasst werden, das anhand einer Eingabe eine dazugehörige Ausgabe erzeugt. Die Eingabe bildet dabei die gewichtete Summe der mit dem Neuron aus der vorherigen Schicht verbundenen Neuronen, wobei die Gewichtung durch die jeweiligen Kantengewichte festgelegt wird. Diese Summe wird durch Aktivierungsfunktion des Neurons auf die Ausgabe abgebildet, welche dann an alle verbundenen Neuronen der Folgeschicht weitergegeben wird.

Das gesamte Netzwerk liefert damit eine Abbildung F , indem man für vorgegebene Eingaben a^0 für die Neuronen der ersten Schicht die Ausgaben $a^S = F(a^0)$ der Neuronen der letzten Schicht ermittelt.

Ein neuronales Netzwerk mit $(S + 1)$ Schichten wird durch die folgenden Größen beschrieben:

- $N_s \in \mathbb{N}$, Anzahl der Neuronen in der Schicht $s = 0, \dots, S$.
- $W^s \in \mathbb{R}^{N_s \times N_{s-1}}$, Gewichte der Kanten zwischen Schicht $(s - 1)$ und Schicht s , $s = 1, \dots, S$.
- $b^s \in \mathbb{R}^{N_s}$, Bias-Vektor, der die gewichtete Summe der Eingaben verschiebt, siehe unten, $s = 1, \dots, S$.
- $\sigma^s: \mathbb{R}^{N_s} \rightarrow [0, 1]^{N_s}$, Aktivierungsfunktionen der Neuronen in der s -ten Schicht, $s = 1, \dots, S$.

Um zu gegebenen Eingabedaten $a^0 \in \mathbb{R}^{N_0}$ die Ausgabe $a^S \in \mathbb{R}^{N_S}$ zu bestimmen,

berechnen wir iterativ

$$z^s = W^s a^{s-1} + b^s \in \mathbb{R}^{N_s}, \quad a^s = \sigma^s(z^s) \in \mathbb{R}^{N_s}, \quad s = 1, \dots, S.$$

Dieser Vorgang, eine Eingabe a^0 durch das Netzwerk zu schicken wird *feedforward* genannt. Zur Eingabe a^0 erhält man also vom Netzwerk die Ausgabe $F(a^0) = a^S$.

Training des Netzwerks Ein Teil der zur Verfügung stehenden Daten (z.B. 75%) verwendet man für das Training, den Rest zur Validierung des Modells. Ziel ist es anhand von K bekannten Datenpaaren $T = \{(a^{0,k}, \hat{a}^{S,k} \in \mathbb{R}^{N_0} \times \mathbb{R}^{N_S} : k = 1, \dots, K\}$, den *Trainingsdaten*, die Gewichte W^s und den Bias b^s in gewissem Sinne optimal zu wählen.

Dazu definiert man eine Fehlerfunktion $c: \mathbb{R}^{N_S} \times \mathbb{R}^{N_S} \rightarrow (0, \infty)$, die eine Ausgabe der Netzwerks mit einer gewünschten Ausgabe vergleicht. Für eine gegebene Eingabe $a^0 \in \mathbb{R}^{N_0}$ und eine zugehörige gewünschte Ausgabe $\hat{a}^S \in \mathbb{R}^{N_S}$ gibt $c(F(a^0), \hat{a}^S)$ den Fehler der Ausgabe des Netzwerks an.

Damit kann der Fehler definiert werden, der zu einem Teil $B \subset T$ des Trainingsdatensatzes gehört:

$$C^B = \sum_{(a^0, \hat{a}^S) \in B} c(F(a^0), \hat{a}^S) \quad (1)$$

Da die *feedforward*-Funktion F von den Gewichten W^s und den Bias-Vektoren b^s abhängt, ist auch der Fehler $C^B(\mathbf{W}, \mathbf{b})$ eine Funktion in $\mathbf{W} = (W^s)_{s=1, \dots, S}$ und $\mathbf{b} = (b^s)_{s=1, \dots, S}$.

Das Training des neuronalen Netzwerks besteht nun darin, die Gewichte und Bias-Vektoren als approximativen Minimierer des Fehlers $C^T(\mathbf{W}, \mathbf{b})$ zu bestimmen, wobei ein stochastisches Gradientenabstiegsverfahren für C^T verwendet wird.

Dazu wählen wir eine Schrittweite $\eta > 0$ sowie Startwerte $\mathbf{W}^0 = (W^{s,0})_{s=1, \dots, S}$ und $\mathbf{b}^0 = (b^{s,0})_{s=1, \dots, S}$. Die Schrittweite η wird auch als *Lernrate* bezeichnet. Außerdem wird eine *Batch-Größe* $k \in \{1, \dots, K\}$ gewählt.

In jedem Schritt $i = 1, 2, \dots$, des Verfahrens wird jeweils eine zufällig gewählte k -elementige Teilmenge $B^i \subset T$ der Trainingsdaten, ein sogenannter *Batch*, betrachtet. Dann werden \mathbf{W}^i und \mathbf{b}^i jeweils in Richtung des negativen Gradienten $\nabla_{\mathbf{W}} C^B(\mathbf{W}^{i-1}, \mathbf{b}^{i-1})$ und $\nabla_{\mathbf{b}} C^B(\mathbf{W}^{i-1}, \mathbf{b}^{i-1})$ aktualisiert.

Anwendung auf die Kästchenklassifikation Für unser Problem verwenden wir ein Netz mit zwei Schichten, also $S = 1$. Wir gehen davon aus, dass alle Kästchen

als Graubilder in einer Größe von 40×40 Pixeln vorliegen. Das Netzwerk hat $N_0 = 1600 = 40 \cdot 40$ Eingabeneuronen, für jedes Pixel eins. Außerdem gibt es für die beiden Klassifikationen “angekreuzt” und “nicht angekreuzt” jeweils ein Ausgabeneuron, also $N_1 = 2$.

Eine Variante des Lernalgorithmus für das Kästchenproblem sieht folgendermaßen aus:

- Initialisiere Gewichtsmatrix $\mathbf{W}^0 = W^{1,0} \in \mathbb{R}^{N_1 \times N_0}$ und Bias-Vektor $\mathbf{b}^0 = b^{1,0} \in \mathbb{R}^{N_1}$ mit $\mathcal{N}(0, 1)$ (Standardnormalverteilung) verteilten Werten. Teile die Gewichte zusätzlich durch $\sqrt{N_0}$. Wähle Batch-Größe k und Anzahl Iterationen E (Epochen).
- Die Grauwerte der Eingabepixel werden normiert auf $[0, 1]$ und in einen Vektor der Länge N_0 gespeichert.
- Für jede Epoche $i = 1, 2, \dots, E$ wähle eine k -elementige Teilmenge $B_i \subset T$

Für jedes Datenpaar $(a^0, \hat{a}^1) \in B^i$ bestimme die Gradienten $\nabla_{\mathbf{W}} C_{\{(a^0, \hat{a}^1)\}}(\mathbf{W}^{i-1}, \mathbf{b}^{i-1})$ und $\nabla_{\mathbf{b}} C_{\{(a^0, \hat{a}^1)\}}(\mathbf{W}^{i-1}, \mathbf{b}^{i-1})$.

Anschließend aktualisiere W und b mit Mittelwerten dieser Gradienten.

$$\mathbf{b}^i = \mathbf{b}^{i-1} - \frac{\eta}{k} \sum_{\{(a_0, \hat{a}_1)\} \in B} \nabla_{\mathbf{b}} C_{\{(a_0, \hat{a}_1)\}}(\mathbf{W}^{i-1}, \mathbf{b}^{i-1})$$

$$\mathbf{W}^i = \mathbf{W}^{i-1} - \frac{\eta}{k} \sum_{\{(a_0, \hat{a}_1)\} \in B} \nabla_{\mathbf{W}} C_{\{(a_0, \hat{a}_1)\}}(\mathbf{W}^{i-1}, \mathbf{b}^{i-1})$$

- Die Ausgabe $a^1 \in \mathbb{R}^2$ kodiert die gesuchte Information. Dabei ist für bekannte Daten der Vektor $(1, 0)^\top$ ein leeres und $(0, 1)^\top$ ein angekreuztes Kästchen. Für unbekannte Daten ist der größere der beiden Einträge ausschlaggebend.
- Beachten Sie, dass das Training einmalig passieren soll und das trainierte Netzwerk für verschiedene Programmabläufe wiederverwendet wird. Überlegen Sie sich deshalb eine Methode, das Netzwerk zu speichern und zu laden.

Konkrete Wahlen für das Kästchenproblem Viele Aktivierungsfunktionen und Kostenfunktionen sind möglich. Hier ein funktionierender Vorschlag:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \text{ (komponentenweise Sigmoid Aktivierungsfunktion)}$$

$$c(a, y) = \sum_{j=1}^{N_S} -y_j \log(a_j) - (1 - y_j) \log(1 - a_j), \text{ (Cross-Entropy-Cost)}$$

$$\frac{\partial c}{\partial z}(\sigma(z), y) = (\sigma(z) - y) \frac{\sigma'(z)}{\sigma(z)(1 - \sigma(z))}, \text{ (komponentenweise)}$$

$$k = 50,$$

$$E \geq 30,$$

$$\eta = 0.1$$

Aufgabe 4 (Statistiken)

Zu einem gescannten Bild eines Bogens werden nun automatisch die angekreuzten Antworten ermittelt. Schreiben Sie Funktionen, die diese Antworten akkumulieren und interessante Statistiken darüber ausgeben können.

Man soll die Analyse über die Konsole starten können, indem man einen Pfad zu einer Sammlung von gescannten Bögen angibt. Nach der Analyse werden in der Konsole eine Zusammenfassung der interessantesten Statistiken ausgegeben.

Literatur

- [1] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.