Aufgabe 23 (Künstliches neuronales Netzwerk zur Klassifizierung handschriftlicher Ziffern)
Auf der Webseite zur Vorlesung[1] findet sich in `mnist_ann.zip` der MNIST-Datensatz und entsprechende Python-Programme zum Trainieren eines künstlichen neuronalen Netzwerks.

(a) Machen Sie sich mit der `Network`-Klasse in `network.py` vertraut und vervollständigen Sie den Code an den zwei (mit `#### ... ####`) angegebenen Stellen. [2 Punkte]

(b) Lassen Sie nun `run_training.py` zum Trainieren des Netzwerks anhand des MNIST-Datensatzes laufen. Wie verändert sich der Klassifizierungserfolg, wenn Sie die Anzahl der Zellen in der versteckten Schicht von 30 auf 5 reduzieren, oder die versteckte Schicht ganz weglassen? [2 Punkte]
Hinweis: Die Ausgabe beim Training sollte in etwa so aussehen:

```
Epoch 1: 9129 / 10000 correct
Epoch 2: 9218 / 10000 correct
...
Epoch 12: 9453 / 10000 correct
```

D.h. nach der ersten "Epoche" (Durchlauf durch den gesamten Trainingsdatensatz) werden bereits 9129 der 10000 Ziffern im (unabhängigen) Testdatensatz korrekt klassifiziert. Aufgrund der zufälligen Initialisierung des Netzwerks weichen die konkreten Werte bei jedem Lauf etwas ab.

(c) Verwenden Sie nun das Netzwerk, um mindestens zwei Ihrer eigenen handschriftlichen Ziffern zu klassifizieren. [2 Punkte]
Hinweis: Scannen oder fotografieren Sie Ihre Handschrift ab, wählen Sie dann einen Ausschnitt mit einer einzelnen Ziffer in dem entsprechenden Bild aus, z.B. mittels Gimp (https://www.gimp.org) oder einem ähnlichen Bildbearbeitungsprogramm, und skalieren Sie anschließend den Ausschnitt auf 28×28 Pixel. In der Datei `image_tools.py` findet sich die Hilfsfunktion `read_image_as_matrix(filename)`, mit der Sie das Bitmap-Bild Ihrer Ziffer als NumPy-Matrix einlesen können. Die NumPy-Methode `flatten()` konvertiert diese Matrix in einen Vektor `x` mit $28^2 = 784$ Einträgen. Die Klassifizierung durch das Netzwerk erhalten Sie schließlich mittels `np.argmax(net.feedforward(x))`.

Aufgabe 24 (Cross-entropy Kostenfunktion)
Die Cross-entropy Kostenfunktion für ein einzelnes Trainingspaar (x, y) lautet

$$C = -\sum_j \Big[y_j \log\big(a_j^L\big) + \big(1 - y_j\big) \log\big(1 - a_j^L\big) \Big], \tag{1}$$

wobei a^L den Ausgabevektor des Netzwerks bezeichnet (Ausgabeschicht L) und $\log$ den natürlichen Logarithmus. Das Netzwerk verwende die Sigmoid-Aktivierungsfunktion $\sigma(z) = 1/(1 + \mathrm{e}^{-z})$ in allen Schichten.

(a) Verifizieren Sie zunächst den Zusammenhang

$$\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big). \tag{2}$$

[1 Punkt]

[1] https://tu-dresden.de/mn/math/wir/mendl/studium/courses/mosim_2018_sose

(b) Im Folgenden soll die spezielle Form (1) der Cross-entropy Kostenfunktion begründet werden. Zur verbesserten "Lernfähigkeit" sollte sich der Faktor $\sigma'(z_j^L)$ in der Hilfsgröße $\delta_j^L = \partial C/\partial z_j^L$ herauskürzen, und δ_j^L sollte umso größer sein, je stärker a^L von y abweicht, was den Ansatz $\delta_j^L = a_j^L - y_j$ nahelegt. Leiten Sie hieraus und aus der BP1-Gleichung

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\,\sigma'(z_j^L)$$

sowie Gl. (2) eine Differentialgleichung für C (als Funktion von a^L) her, und zeigen Sie, dass diese von Gl. (1) gelöst wird. [3 Punkte]

(c) Eine Verallgemeinerung von Gl. (1) ist die Cross-entropy zweier diskreter Wahrscheinlichkeitsverteilungen p und q (wobei wir $p_i > 0$ und $q_i > 0$ für alle i annehmen):

$$H(p,q) = -\sum_i p_i \log(q_i).$$

Diese lässt sich auch als $H(p,q) = H(p) + D_{\mathsf{KL}}(p \parallel q)$ schreiben, wobei

$$H(p) = -\sum_i p_i \log(p_i)$$

die Entropie der Verteilung p ist, und

$$D_{\mathsf{KL}}(p \parallel q) = \sum_i p_i \log \frac{p_i}{q_i}$$

die Kullback-Leibler-Divergenz. Beweisen Sie die sogenannte Gibbs-Ungleichung: $D_{\mathsf{KL}}(p \parallel q) \geq 0$, mit Gleichheit genau für $p = q$. [2 Punkte]

Hinweis: Für alle $x > 0$ gilt $-\log(x) \geq (1-x)$, mit Gleichheit genau für $x = 1$. (Diese Aussage können Sie als bekannt voraussetzen.) Wenden Sie diese Aussage auf $\log(p_i/q_i) = -\log(q_i/p_i)$ an, und nutzen Sie die Normalisierung der Wahrscheinlichkeitsverteilungen ($\sum_i p_i = 1$, $\sum_i q_i = 1$) aus.

Tutoraufgabe (Universalität künstlicher neuronaler Feedforward-Netze)
Künstliche neuronale Feedforward-Netze sind *universell*, d.h. zu jeder kompakten Teilmenge $K \subset \mathbb{R}^n$ und stetigen Funktion $f : K \to \mathbb{R}$ sowie Fehlerschranke $\epsilon > 0$ existiert ein Feedforward-Netz $\mathcal{N}$, so dass $|\mathcal{N}(x) - f(x)| < \epsilon$ für alle $x \in K$ gilt, wobei $\mathcal{N}(x)$ die Ausgabe des Netzes bezeichnet (und die Sigmoid-Aktivierungsfunktion in der Ausgabeschicht weggelassen wird, um z.B. auch negative Ausgabewerte zu ermöglichen).[2] Ziel dieser Aufgabe ist die Skizze eines konstruktiven Beweises der Universalität.

(a) Es sei zunächst $n = 1$. Zeigen Sie durch geeignete Wahl von $w, b \in \mathbb{R}$, dass eine einzelne Nervenzelle mit Ausgabe $\sigma(wx + b)$ eine Stufenfunktion $x \mapsto \Theta(x - s)$ nachbilden kann. Wie erhält man durch Linearkombination von Stufenfunktionen eine Rechteckfunktion bzw. beliebige Treppenfunktion? Konstruieren Sie hieraus ein Netzwerk zur Approximation einer stetigen Funktion $f : K \to \mathbb{R}$, wobei o.B.d.A. $K = [0,1]$ angenommen werden kann.

(b) Zur Verallgemeinerung auf beliebiges $n \geq 1$ konstruieren Sie mittels (a) zunächst eine Summe von Rechteckfunktionen in jeder Koordinatenrichtung, und unter Verwendung eines Schwellenwerts eine charakteristische Funktion auf einem beliebigen Hyperrechteck. Approximieren Sie nun eine Funktion $f : K \to \mathbb{R}$ durch eine Linearkombination charakteristischer Funktionen.

[2] siehe z.B. G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems* 2, 303–314 (1989) [http://www.dartmouth.edu/~gvc/Cybenko_MCSS.pdf] und K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.* 2, 359–366 (1989).