Aufgabe 25 (Optimiertes Trainieren neuronaler Netze)
In `improved_learning.zip` auf der Vorlesungs-Webseite findet sich wiederum der MNIST-Datensatz sowie verbesserte und erweiterte Python-Programme zum Trainieren eines künstlichen neuronalen Netzwerks (siehe `network2.py`). Die quadratische oder Cross-entropy Kostenfunktion kann nun mittels `Network(sizes, cost=QuadraticCost)` bzw. `Network(sizes, cost=CrossEntropyCost)` ausgewählt werden. Hierfür werden zwei entsprechende Hilfsklassen in `network2.py` definiert, die die Kostenfunktion und deren Ableitung zur Berechnung von δ^L als statische Methoden zur Verfügung stellen. Die Klassenmethode `SGD` kann jetzt den Trainingsfortschritt mitprotokollieren.

(a) Machen Sie sich mit der Implementierung vertraut, und lassen Sie `run_training.py` einmal mit quadratischer und einmal mit Cross-entropy Kostenfunktion laufen. Vergleichen Sie anschließend die beiden Programmläufe, indem Sie die relative "accuracy" als Funktion der Trainingsepoche visualisieren, sowohl bezüglich des Test- als auch des Trainingsdatensatzes. [3 Punkte]

Hinweis: Für die quadratische Kostenfunktion eignet sich die Lernrate $\eta = 3$, und für Cross-entropy $\eta = 0.5$. Die Variablen `test_accuracy` und `training_accuracy` speichern die Anzahl der korrekt klassifizierten Ziffern im Test- bzw. Trainingsdatensatz für jede Epoche. Zur Reproduzierbarkeit eines Programmlaufs wird der NumPy-Zufallszahlengenerator mittels `np.random.seed(...)` in `run_training.py` initialisiert; Sie können natürlich auch einen anderen Startwert als den voreingestellten wählen.

(b) Untersuchen Sie nun ein Overfitting-Szenario, indem Sie den Trainingsdatensatz künstlich auf die ersten 1000 Ziffern reduzieren und dann das Netzwerk neu trainieren (mit Cross-entropy Kostenfunktion). Verwenden Sie hierzu 400 anstatt 30 Trainingsepochen. Plotten Sie anschließend den Verlauf der Kostenfunktion (Variable `training_cost`) sowie den Klassifizierungserfolg als Validierung (Variable `test_accuracy`), jeweils ab Trainingsepoche 100. Woran lässt sich Overfitting erkennen? [2 Punkte]

(c) Quantifizieren Sie schließlich die Auswirkung von L2-Regularisierung, indem Sie das Experiment in (b) mit Regularisierungsparameter $\lambda = 0.1$ anstatt $\lambda = 0$ wiederholen. Der entsprechende Parameter im Python-Code lautet `lmbda`.[1] Wie ändert sich der Verlauf der Kostenfunktion bzw. des Klassifizierungserfolgs? [1 Punkt]

Aufgabe 26 (Instabile Gradienten bei tiefen neuronalen Netzen)
Wiederholtes Anwenden der (BP2)-Gleichung des Backpropagation-Algorithmus führt auf

$$\delta^\ell = \mathrm{diag}\big(\sigma'(z^\ell)\big) \cdot \big(w^{\ell+1}\big)^T \cdot \mathrm{diag}\big(\sigma'(z^{\ell+1})\big) \cdot \big(w^{\ell+2}\big)^T \cdots \mathrm{diag}\big(\sigma'(z^L)\big) \cdot \nabla_{a^L} C.$$

Zur Vereinfachung nehmen wir an, dass jede Schicht aus einem einzelnen Neuron besteht, also w^ℓ, b^ℓ, z^ℓ, a^ℓ jeweils reelle Zahlen sind und obige Gleichung die Form $\delta^\ell = \sigma'(z^\ell) \cdot w^{\ell+1} \cdot \sigma'(z^{\ell+1}) \cdot w^{\ell+2} \cdots \sigma'(z^L) \cdot \nabla_{a^L} C$ annimmt.

(a) Verifizieren Sie, dass die Ableitung der Sigmoid-Aktivierungsfunktion $\sigma(z) = 1/(1+\mathrm{e}^{-z})$ stets im Intervall $[0, 1/4]$ liegt, d.h. $0 \le \sigma'(z) \le \frac{1}{4}\ \forall z \in \mathbb{R}$ gilt. [2 Punkte]

Bei einer typischen Wahl "nicht zu großer" Gewichte w^ℓ (d.h. konkret $|w^\ell| < 4$) ist also jeder Faktor $w^\ell \sigma'(z^\ell)$ in der obigen Darstellung von $\delta^{\ell'}$ ($\ell' < \ell$) betragsmäßig kleiner als 1, was zu einer exponentiellen Abnahme des Gradienten (und entsprechend der effektiven Lernrate) mit wachsender Netzwerktiefe führt. Umgekehrt kann man im Fall $|w^\ell| \ge 4$ überlegen, für welche Werte von $a^{\ell-1}$ überhaupt $|w^\ell \sigma'(z^\ell)| \equiv |w^\ell \sigma'(w^\ell a^{\ell-1} + b^\ell)| \ge 1$ zutrifft. Auflösen dieser Bedingung nach $a^{\ell-1}$ ergibt, dass $a^{\ell-1}$ in dem (um den Ursprung zentrierten) Intervall der Länge

$$\frac{2}{|w^\ell|} \log\left(\frac{|w^\ell|}{2} \big(1 + \sqrt{1 - 4/|w^\ell|}\big) - 1 \right)$$

liegen muss. (Ein Nachweis dieser Aussage ist hier nicht gefordert.)

(b) Visualisieren Sie diese Intervalllänge als Funktion von $|w^\ell|$, und bestimmen Sie dessen Maximum numerisch. Ist es also sehr wahrscheinlich, dass $a^{\ell-1}$ tatsächlich in dem Intervall liegt? [2 Punkte]

Es bezeichne $q^\ell = |w^\ell \sigma'(z^\ell)|$. Obwohl also auch $q^\ell \ge 1$ möglich ist, bleibt ein grundlegendes Problem beim Trainieren tiefer Netze der große Unterschied in den $|\delta^\ell| = |\sigma'(z^\ell)| \cdot q^{\ell+1} \cdot q^{\ell+2} \cdots q^L \cdot |\nabla_{a^L} C|$ für verschiedene ℓ, und den damit einhergehenden stark unterschiedlichen Gradienten. Zur Abschätzung modellieren wir die q^ℓ als unabhängige, identisch verteilte (i.i.d.) Zufallsvariablen, und schreiben

$$Q^\ell := q^{\ell+1} \cdot q^{\ell+2} \cdots q^L = \mathrm{e}^{\log(q^{\ell+1}) + \cdots + \log(q^L)}.$$

(c) Es bezeichne μ den Mittelwert und s^2 die Varianz von $\log(q^\ell)$ für alle ℓ. Geben Sie hiermit den Mittelwert und die Varianz von Q^ℓ an. [2 Punkte]

Hinweis: Sie können von einer logarithmischen Normalverteilung[2] für Q^ℓ ausgehen, d.h. $\log(q^{\ell+1}) + \cdots + \log(q^L)$ als normalverteilte Zufallsvariable betrachten. Die Varianz einer Summe unabhängiger Zufallsvariablen ist die Summe der Varianzen.

[1] `lambda` ist ein Schlüsselwort in Python zur Definition anonymer Funktionen und kann deshalb nicht als Variablenname verwendet werden.
[2] https://en.wikipedia.org/wiki/Log-normal_distribution

Tutoraufgabe (Backpropagation für komplexwertige Netze)
In dieser Aufgabe sollen die Gleichungen (BP1) – (BP4) des Backpropagation-Algorithmus auf komplexwertige neuronale Netze verallgemeinert werden, d.h. w^ℓ, b^ℓ, z^ℓ, a^ℓ sind jetzt komplexe Matrizen bzw. Vektoren. Wir nehmen an, dass die Aktivierungsfunktion in einer offenen Menge $U \subset \mathbb{C}$ holomorph ist, was ja insbesondere auf die Sigmoid-Funktion $\sigma(z) = 1/(1+\mathrm{e}^{-z})$ zutrifft. Wie im reellen Fall verwenden wir die Hilfsgröße

$$\delta_j^\ell := \frac{\partial C}{\partial z_j^\ell},$$

wobei jetzt allerdings die Ableitungsoperatoren als *Wirtinger*- oder *Dolbeault*-Operatoren aufgefasst werden:

$$\frac{\partial}{\partial z} = \frac{1}{2}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right), \quad \frac{\partial}{\partial \overline{z}} = \frac{1}{2}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right)$$

mit $z = x + iy$, $x, y \in \mathbb{R}$. Außerdem benötigen wir noch die folgende Kettenregel, die sich durch direktes Nachrechnen verifizieren lässt:

$$\frac{\partial}{\partial z}(g \circ f) = \left(\frac{\partial g}{\partial w} \circ f\right) \cdot \frac{\partial f}{\partial z} + \left(\frac{\partial g}{\partial \overline{w}} \circ f\right) \cdot \frac{\partial \overline{f}}{\partial z}.$$

Die Kostenfunktion C ist weiterhin reellwertig (so dass deren Minimierung überhaupt Sinn ergibt), und somit nicht komplex differenzierbar.

Hinweis: Die $\mathbb{C}$-lineare Multiplikation $w \mapsto z \cdot w$ mit einer vorgegebenen komplexen Zahl $z = x + iy$ lässt sich durch Identifikation $\mathbb{C} \cong \mathbb{R}^2$ auch als $\mathbb{R}$-lineare Abbildung auffassen. Anwendung auf $1 = (1,0)$ und $i = (0,1)$ ergibt die Matrixdarstellung

$$A_z = (z|iz) = \begin{pmatrix} x & -y \\ y & x \end{pmatrix}. \tag{1}$$

Umgekehrt ist eine reelle 2×2-Matrix $A = (z_1|z_2)$ mit Spaltenvektoren $z_1, z_2 \in \mathbb{C}$ also von der Form (1), wenn $z_2 = iz_1$ bzw. $z_1 + iz_2 = 0$.
Eine Funktion $f : U \subset \mathbb{C} \to \mathbb{C}$ ist bekanntlich genau dann in $z \in U$ komplex differenzierbar, wenn

$$f(w) = f(z) + f'(z) \cdot (w - z) + o(w - z) \qquad (w \to z). \tag{2}$$

Identifizieren wir $\mathbb{C} \cong \mathbb{R}^2$, dann ist f (als Funktion $U \subset \mathbb{R}^2 \to \mathbb{R}^2$) genau dann reell differenzierbar, wenn

$$f(w) = f(z) + Df(z) \cdot (w - z) + o(w - z) \qquad (w \to z).$$

Hierbei ist $Df(z) \in \mathbb{R}^{2\times 2}$ die Jakobi-Matrix. Die stärkere Bedingung "komplex differenzierbar" bedeutet also, dass das Matrix-Vektor-Produkt $Df(z) \cdot (w - z)$ als Multiplikation zweier komplexer Zahlen aufgefasst werden kann. Zerlegen wir $Df = (\partial_x f | \partial_y f)$, dann ist dies laut obiger Diskussion äquivalent zu den Cauchy-Riemann'schen Differentialgleichungen

$$\frac{\partial f}{\partial \overline{z}} \equiv \frac{1}{2}\left(\partial_x f + i\partial_y f\right) = 0,$$

wobei $\partial/\partial\overline{z}$ der (konjugierte) Wirtinger-Operator ist. Die komplexe Ableitung ist dann $f'(z) = \partial f(z)/\partial z$.
Als direkte Folgerung erhält man für f komplex differenzierbar in z:

$$\frac{\partial \overline{f}}{\partial z} = \frac{1}{2}\left(\partial_x \overline{f} - i\partial_y \overline{f}\right) = \overline{\frac{1}{2}\left(\partial_x f + i\partial_y f\right)} = \overline{\left(\frac{\partial f}{\partial \overline{z}}\right)} = 0. \tag{3}$$

(a) Zeigen Sie, dass (formal identisch zum reellwertigen Fall)

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L), \tag{BP1}$$

wobei $\partial/\partial a_j^L$ die Wirtinger-Ableitung bezüglich a_j^L ist und σ' die komplexe Ableitung der Aktivierungsfunktion. Wie lautet $\partial C/\partial a_j^L$ für die quadratische Kostenfunktion $C = \frac{1}{2}\|y - a^L\|^2$?

(b) Leiten Sie (wiederum formal identisch zum reellwertigen Fall) den Zusammenhang

$$\delta^\ell = \mathrm{diag}\big(\sigma'(z^\ell)\big) \cdot \big(w^{\ell+1}\big)^T \cdot \delta^{\ell+1} \tag{BP2}$$

für alle $\ell < L$ her, wobei die Ableitung σ' komponentenweise an den Einträgen von z^ℓ ausgewertet wird.

(c) Verifizieren Sie ebenso

$$\frac{\partial C}{\partial b_j^\ell} = \delta_j^\ell \tag{BP3}$$

sowie

$$\frac{\partial C}{\partial w_{jk}^\ell} = a_k^{\ell-1}\delta_j^\ell. \tag{BP4}$$

(d) Wie kann man nun den Gradienten der Kostenfunktion bezüglich des Real- und Imaginärteils von w_{jk}^ℓ bzw. b_j^ℓ erhalten?