Jun.-Prof. Dr. Christian Mendl Abgabe: 16. Januar 2019

Aufgabe 11 (Optimiertes Trainieren neuronaler Netze anhand des CIFAR-10 Datensatzes)
In `improved_learning.zip` auf der Vorlesungs-Webseite finden sich verbesserte und erweiterte Python-Programme zum Trainieren eines künstlichen neuronalen Netzwerks.[1] Zusätzlich zur Sigmoid-Aktivierungsfunktion werden sogenannte "Rectified Linear Units" (ReLU) unterstützt, d.h. Zellen mit Aktivierungsfunktion $z \mapsto \max(0, z)$. Die modulare Implementierung realisiert Aktivierungsfunktionen formal als eigene Schichten; somit besteht eine logische Schicht in dem Code aus einer affinen Schicht zur Berechnung von $z_j = \sum_k w_{jk} x_k + b_j$, gefolgt von einer Schicht zur punktweisen Berechnung von $a_j = \sigma(z_j)$, siehe `layers.py`.

Der CIFAR-10 Datensatz[2] ist eine populäre Referenz für Machine Learning-Forschungsprojekte (konzeptionell ähnlich zu MNIST). Er besteht aus 60000 32×32 Farbbildern, eingeteilt in 10 Klassen. Aufgrund der RGB-Farbkanäle beträgt die Eingabedimension für ein neuronales Netzwerk somit $3 \cdot 32^2 = 3072$. Sie erhalten den Datensatz wie in `get_cifar10.md` beschrieben.

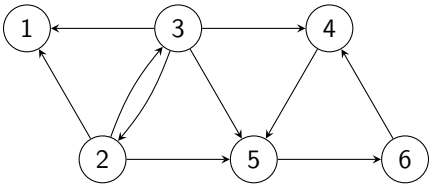
(a) In `run_overfitting.py` ist der Trainingsdatensatz bewusst auf nur 2500 Trainingspaare reduziert. Passen Sie den Typ der Aktivierungsfunktion, den Skalierungsparameter für die anfängliche Initialisierung der Gewichtsmatrizen sowie die Lernrate in `run_overfitting.py` so an, dass Sie mindestens 60% Klassifikationsgenauigkeit beim Trainingsdatensatz erreichen. Woran lässt sich Overfitting anhand des Ausgabeplots erkennen? [2 Punkte]

(b) Das Adam-Optimierungsverfahren[3] verwendet eine adaptive, pro Parameter individuelle Lernrate, um Schwachstellen des gewöhnlichen SGD-Verfahrens zu umgehen. Implementieren Sie das Adam-Verfahren in `optim.py`, und testen Sie Ihre Implementierung durch Aufruf von `test_optim.py`. [2 Punkte]

(c) Versuchen Sie als Herausforderung, einen möglichst hohen Klassifizierungserfolg beim CIFAR-10 Datensatz zu erreichen, indem Sie die Netzwerk-Parameter und -Architektur (Typ der Aktivierungsfunktion, Anzahl versteckter Schichten und Anzahl Zellen in jeder Schicht, ...) bzw. Hyperparameter in `run_training.py` geschickt wählen. [2 Punkte]

Hinweis: Sie können den Trainingsdatensatz durch die Modifikation `get_CIFAR10_data(num_training=20000)` verkleinern, um die Trainingszeiten zu verkürzen. Für die Teilaufgabe sollten Sie mindestens 40% Genauigkeit beim Testdatensatz erreichen.

Aufgabe 12 (Hyperlink-Struktur)
Wir untersuchen die Hyperlink-Struktur des folgenden Modell-Internets:



(a) Stellen Sie die Hyperlink-Matrix $H = (h_{ij})$ mit Einträgen

$$h_{ij} = \begin{cases} 1/|S_i| & S_i \text{ besitzt einen Link auf } S_j \\ 0 & \text{sonst} \end{cases}$$

auf, wobei $|S_i|$ die Anzahl der ausgehenden Links der Seite S_i bezeichnet. [2 Punkte]

(b) Berechnen Sie (symbolisch oder numerisch mit Angabe des entsprechenden Codes) die Eigenwerte von H. Wie viele Eigenwerte λ mit $|\lambda| = 1$ gibt es? [2 Punkte]

(c) Begründen Sie, dass die Iterationsvorschrift

$$\pi^{k+1} = H^T \pi^k, \quad k = 0, 1, \ldots$$

im Allgemeinen nicht konvergiert, indem Sie als anfängliche Zustandsverteilung $\pi^0 = (0, 0, 0, 0, 0, 1)^T$ wählen. [2 Punkte]

[1] basierend auf Stanford CS231n: Convolutional Neural Networks for Visual Recognition (http://cs231n.stanford.edu)
[2] https://www.cs.toronto.edu/~kriz/cifar.html
[3] D. P. Kingma, J. Ba. Adam: A Method for Stochastic Optimization. ICLR 2015 (https://arxiv.org/abs/1412.6980)

Tutoraufgabe 6 (Instabile Gradienten bei tiefen neuronalen Netzen)
Wiederholtes Anwenden der Gleichungen des Backpropagation-Algorithmus führt auf

$$\nabla_{a^\ell} C_{x,y} = \left(w^{\ell+1}\right)^T \cdot \operatorname{diag}\left(\sigma'(z^{\ell+1})\right) \cdot \left(w^{\ell+2}\right)^T \cdots \operatorname{diag}\left(\sigma'(z^L)\right) \cdot \nabla_{a^L} C_{x,y}.$$

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 $\nabla_{a^\ell} C_{x,y} = w^{\ell+1} \cdot \sigma'(z^{\ell+1}) \cdot w^{\ell+2} \cdots \sigma'(z^L) \cdot \nabla_{a^L} C_{x,y}$ 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.

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 $\nabla_{a^{\ell'}} C_{x,y}$ ($\ell' < \ell$) 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

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

liegen muss.

(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?

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 $|\nabla_{a^\ell} C_{x,y}| = q^{\ell+1} \cdot q^{\ell+2} \cdots q^L \cdot |\nabla_{a^L} C|$ 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.
Hinweis: Sie können von einer logarithmischen Normalverteilung für Q^ℓ ausgehen, d.h. $\log(q^{\ell+1}) + \cdots + \log(q^L)$ als normalverteilte Zufallsvariable betrachten (siehe z.B. https://en.wikipedia.org/wiki/Log-normal_distribution). Die Varianz einer Summe unabhängiger Zufallsvariablen ist die Summe der Varianzen.

Bemerkung: Der Adam-Optimierungsalgorithmus umgeht das Problem der stark unterschiedlichen Gradienten größtenteils, indem die Lernrate adaptiv und für jeden Netzwerkparameter separat gewählt wird.