NNW-Übung 7

1 PyTorch `torch.nn.Module`: Klassen für Neuronale Netzwerke

Bisher haben wir die Berechnungen einzelner Neuronen noch selbst programmiert. Dafür gibt es in PyTorch aber vorgefertigte Klassen, die Schichten (engl. *layers*) von Neuronen realisieren, z.B. für eine *fully connected Schicht* von Neuronen *ohne* Aktivierungsfunktion die Klasse torch.nn.Linear.

Diese Schichten sind Sub-Klassen der Klasse `torch.nn.Module` – und können zu neuen Sub-Klassen der Klasse `torch.nn.Module` kombiniert werden! Wie das geht, lernen Sie jetzt kennen.

a) Im dem im Folgenden genannten Tutorial geht es um eine schwierigere Variante des berühmten *MNIST-Datensatzes*:

   - Der klassische *MNIST-Datensatz* enthält insgesamt 70000 kleine Bilder[1] der handschriftliche Ziffern von 0 bis 9 – also 10 verschiedene Klassen. Die Größe der einzelnen Bilder ist 28x28.
   - Mit heutigen Methoden ist dieses Klassifikationsproblem allerdings schon recht / zu einfach, siehe hier für einige Ergebnisse. Daher wurde (u.a.) der schwieriger zu klassifizierende Datensatz *FashionMNIST* erstellt mit Bildern von 10 Kategorien von Kleidungsstücken – mit gleicher Anzahl und Bildgröße.

b) Lesen Sie das folgende Tutorial in Ruhe durch ... und führen Sie es Abschnitt für Abschnitt selbst aus!
   `https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html`

c) Was bedeuten die beiden Parameter, die an `torch.nn.Linear` übergeben werden?

d) Was ist der Unterschied zwischen `__init__` und `forward()`?

e) Wo wird festgelegt, welcher Output als Input von der nächsten Schicht verwendet wird?

f) Wann wird tatsächlich etwas berechnet?

g) Was ist der Unterschied zwischen:

   - `nn.Flatten`
   - `self.flatten` (im Code des Tutorials)
   - `self.flatten(x)` (im Code des Tutorials)
   - `torch.flatten` und
   - `torch.tensor.flatten`
   - sowie zwischen `torch.Tensor.to` und `torch.nn.Module.to`?

h) Warum ist die Formel bei torch.nn.Linear anders (siehe die Dokumentation), als wir sie (bisher) in der Vorlesung hatten?

2 PyTorch Training

In dem oben genannten Tutorial wurden noch *keine* Daten geladen und das Netz noch *nicht* trainiert. Wie das geht, kommt jetzt.

[1] 60000 zum Training, 10000 zum Testen.

a) Lesen Sie das folgende Tutorial in Ruhe durch ... und führen Sie es Abschnitt für Abschnitt selbst aus!
`https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html`

Beachten Sie:

- Die Zielwerte (targets), die wir in der Vorlesung mit `T` bezeichnen, werden in den PyTorch-Tutorials (leider) mit `y` bezeichnet.
- Die Ausgaben/Vorhersagen des Netzwerks, die wir in der Vorlesung mit `y` bezeichnen, werden in den PyTorch-Tutorials (oft) mit `pred` (predictions) bezeichnet (manchmal, bei Klassifikationsaufgaben, inhaltlich begründet auch als `logits`).
- Es folgen einige Fragen zum Code – bei manchen müssen Sie etwas raten, in der Dokumentation nachsehen oder das unten genannte, weiterführende Tutorial ansehen.

b) Wie viele Dimensionen hat das Daten-Array `X` und was ist in den einzelnen Dimensionen enthalten?

c) Was enthält `y`?

d) Wie viele Outputs liefert die Netzwerk-Klasse? (Wundert Sie hier zunächst etwas?)

e) Was bedeutet der Parameter `lr` bei der Erstellung des Objekts `optimizer`?

f) Was macht `train()`?

g) Was macht `optimizer.step()`?

h) Kennt der `optimizer` die Loss-Funktion?

i) Woher weiß der `optimizer`, welche Variablen er in einem Aktualisierungsschritt ändern soll?

j) Woher kennt `optimizer` den Gradienten der Loss-Funktion bezüglich dieser Variablen?

k) Was macht `optimizer.zero_grad()`? Warum ist das wichtig?

l) Lesen Sie nun auch das folgende, weiterführende Tutorial (das von dem gerade betrachteten Tutorial verlinkt wird) in Ruhe durch ... und führen Sie auch dieses Tutorial Abschnitt für Abschnitt selbst aus!
`https://pytorch.org/tutorials/beginner/basics/optimization_tutorial.html`

m) Optional: Lassen Sie die Gesamtzahl der beim Training angepassten Gewichte berechnen und die Gewichte je Schicht berechnen.

n) Wie groß ist die Accuracy auf den Testdaten bei der im Tutorial realisierten Netzwerk?

o) Ändern Sie das Netzwerk zu einem Single-Layer-Netzwerk. Welche Accuracy erreicht dieses Netzwerk bei ansonsten unveränderten Einstellungen?

p) Ändern Sie den Optimierer auf `torch.optim.Adam` mit dessen Standardeinstellungen (ohne Angabe weiterer Parameter). Welche Accuracies werden jetzt bei den beiden Netzwerke erreicht?

q) Ändern Sie die Daten von FashionMNIST auf MNIST. Welche Accuracies werden jetzt bei den beiden Netzwerke erreicht bei ansonsten unveränderten Einstellungen (weiterhin Verwendung von Adam)?

r) Probieren Sie für die beiden Netzwerke die Funktion `summary` des Packages `torchinfo` aus, das Sie (wahrscheinlich) zunächst installieren müssen.