19. lipnja 2020.

Zavod za elektroniku, mikroelektroniku,
računalne i inteligentne sustave

Duboko učenje

završni ispit

1. (10b) Zadani su niz operacija unaprijednog prolaza i nedovršeni niz operacija unatražnog prolaza:

Unaprijedni prolaz:

$$\begin{aligned}
&\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}, \quad (\boldsymbol{W} \in \mathbb{R}^{m\times n})\\
&\boldsymbol{h}' = \mathrm{ReLU}(\boldsymbol{h})\\
&s = \boldsymbol{w}^\mathsf{T}\boldsymbol{h}'\\
&L = -\ln(\sigma(s))[\![y=1]\!] - \ln(1-\sigma(s))[\![y=0]\!],
\end{aligned}$$

Unatražni prolaz:

$$\begin{aligned}
&\frac{\partial L}{\partial s} = \dots\\
&\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial s}\cdot\dots, \quad \frac{\partial L}{\partial \boldsymbol{h}'} = \dots\\
&\quad\vdots\\
&\frac{\partial L}{\partial (\boldsymbol{W}_{[i,:]})^\mathsf{T}} = \dots, \quad \frac{\partial L}{\partial \boldsymbol{b}} = \frac{\partial L}{\partial \boldsymbol{h}}.
\end{aligned}$$

Zadaci:

a) (1b) Koliko operacija skalarnog zbrajanja i množenja je potrebno za izračunavanje $\boldsymbol{h}$ uz najjednostavniju implementaciju?

b) (2b) Napišite izraz za izračunavanje elementa $\boldsymbol{h}_{[i]}$ preko definicije matričnog množenja i izvedite derivacije $\frac{\partial \boldsymbol{h}_{[i]}}{\partial \boldsymbol{x}_{[j]}}$, $\frac{\partial \boldsymbol{h}_{[i]}}{\partial \boldsymbol{W}_{[k,l]}}$ i $\frac{\partial \boldsymbol{h}_{[i]}}{\partial \boldsymbol{b}_{[j]}}$.

c) (1b) Kojih su dimenzija tenzori $\boldsymbol{w}$, $\frac{\partial \boldsymbol{h}'}{\partial \boldsymbol{h}}$ i $\frac{\partial L}{\partial \boldsymbol{h}'}$ i koliko je njihovih elemenata je uvijek 0?

d) (4b) Dovršite jednadžbe unatražnog prolaza tako da se dobiju derivacije gubitka (jakobijani) po međurezultatima i parametrima bez redundantnih operacija.

e) (2b) Nacrtajte računske grafove unaprijednog i unatražnog prolaza tako da čvorovi predstavljaju ulazne varijable (npr. $\boldsymbol{x}$, $\boldsymbol{w}$) i međurezultate s operacijama (npr. $(\boldsymbol{h}, +)$, $(\boldsymbol{h}', \mathrm{ReLU})$), a bridovi izravne zavisnosti – po uzoru na sliku dolje. Elementarnim operacijama smatrajte matrično množenje, zbrajanje vektora, ReLU i cijelu operaciju računanja gubitka $(s, y) \mapsto L$.

Rješenje:

a) (1b) $(m-1)n + m = mn$ zbrajanja (priznati i $mn + m$), mn množenja

b) (2b)

$$\boldsymbol{h}_{[i]} = \sum_j \boldsymbol{W}_{[i,j]}\boldsymbol{x}_{[j]} + \boldsymbol{b}_{[i]}, \quad \frac{\partial \boldsymbol{h}_{[i]}}{\partial \boldsymbol{x}_{[j]}} = \boldsymbol{W}_{[i,j]}, \quad \frac{\partial \boldsymbol{h}_{[i]}}{\partial \boldsymbol{W}_{[k,l]}} = [\![k=i]\!]\boldsymbol{x}_{[l]}, \quad \frac{\partial \boldsymbol{h}_{[i]}}{\partial \boldsymbol{b}_{[j]}} = [\![i=j]\!] \quad (1)$$

c) (1b) $\dim(\boldsymbol{w}) = m$ (priznati i $(m, 1)$), $\dim\left(\frac{\partial \boldsymbol{h}'}{\partial \boldsymbol{h}}\right) = (m, m)$ i $\dim\left(\frac{\partial L}{\partial \boldsymbol{h}'}\right) = (1, m)$

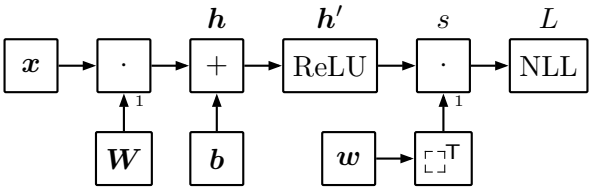
d) (4b):

$$\begin{aligned}
&\frac{\partial L}{\partial s} = \sigma(s) - [\![y=1]\!]\\
&\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial s}\frac{\partial s}{\partial \boldsymbol{h}'} = \frac{\partial L}{\partial s}\boldsymbol{w}^\mathsf{T}, \quad \frac{\partial L}{\partial \boldsymbol{h}'} = \frac{\partial L}{\partial s}\frac{\partial s}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial s}\boldsymbol{h}'^\mathsf{T}\\
&\frac{\partial L}{\partial \boldsymbol{h}} = \frac{\partial L}{\partial \boldsymbol{h}'}\frac{\partial \boldsymbol{h}'}{\partial \boldsymbol{h}} = \frac{\partial L}{\partial \boldsymbol{h}'}\,\mathrm{diag}\big(\mathrm{sgn}\big(\boldsymbol{h}'\big)\big)\\
&\frac{\partial L}{\partial (\boldsymbol{W}_{[i,:]})^\mathsf{T}} = \frac{\partial L}{\partial \boldsymbol{h}}\frac{\partial \boldsymbol{h}}{\partial (\boldsymbol{W}_{[i,:]})^\mathsf{T}} = \frac{\partial L}{\partial \boldsymbol{h}_{[i]}}\frac{\partial \boldsymbol{h}_{[i]}}{\partial (\boldsymbol{W}_{[i,:]})^\mathsf{T}} = \frac{\partial L}{\partial \boldsymbol{h}_{[i]}}\boldsymbol{x}^\mathsf{T}, \quad \frac{\partial L}{\partial \boldsymbol{b}} = \frac{\partial L}{\partial \boldsymbol{h}}\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{b}} = \frac{\partial L}{\partial \boldsymbol{h}}.
\end{aligned}$$

Izvod (0 bodova, traži se u 3. zadatku):

$$L = \ln(\exp(s) - 1) - s[\![y=1]\!] \implies \frac{\partial L}{\partial s} = \frac{\exp(s)}{\exp(s)-1} - [\![y=1]\!] = \sigma(s) - [\![y=1]\!]$$

e) (2b)



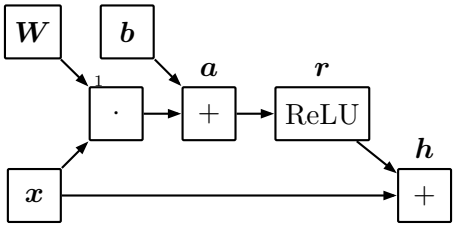
TODO unatražni graf

f) Poboljšanje zadatka: Reći da $\boldsymbol{x}$ nije mini-grupa, nego samo 1 primjer. Tražiti $\frac{\partial \boldsymbol{h}'_{[i]}}{\partial \boldsymbol{h}_{[j]}}$. "skalarno zbrajanje i množenje" može biti dvoznačno – množenje skalara i skalarni produkt. Umjesto crtanja cijelog unatražnog grafa tražiti samo dodjelivanje jakobijana strelicama? Naći bolji način da se kaže da pišu jednadžbe bez redundancije (prepisivanja već napisanog).)

2. (10b) Nadopunite `layers.py` iz druge laboratorijske vježbe razredom koji opisuje modul na slici. Podsjetite se da sučelje `Layer` implementira sljedeće tri metode:

   - `forward(self, inputs)` - vraća izlaz sloja
   - `backward_inputs(self, grads)` - vraća gradijent po ulazu
   - `backward_params(self, grads)` - vraća listu parova (parametar, gradijent po parametru)

   Implementirajte prikazani modul u Pytorchu ili Tensorflowu.



Rješenje 1 numpy():

```
import numpy as np

class ResBlock(Layer):
    def __init__(self, num_inputs, name):
        # 1 bod
        self.name = name
        self.W = np.random.rand(num_inputs, num_inputs)
        self.b = np.random.rand(num_inputs)

    def forward(self, inputs):
        # 2 boda
        # out = in + a = in + relu(h) = in + relu(in*W + b)
        self.inputs = inputs
        self.h = inputs@self.W + self.b
        a = self.h
        a[self.h<0] = 0
        output = a + inputs

    def backward_inputs(self, grads):
        # 2 boda
        # dL/d_in = dL/d_out*d_out/d_in
        #         = dL/d_out*d(in + a)/d_in
        #         = dL/d_out*(d_in/d_in + d_a/d_in)
        #         = dL/d_out(d_in/d_in+d_a/d_h*dh/d_in)
        #         = dL/d_out + dL/d_out*d_a/d_h*d_h/d_in)
        dL_dh = grads
        dL_dh[self.h<=0] = 0

        dL_din = dL_dh@self.W.T
```

```
        return dL_din + grads

    def backward_params(self, grads):
        # 2 boda
        # dL/d_W = dL/d_out*d_out/d_W
        #        = dL/d_out*d(in + a)/d_W
        #        = dL/d_out*(d_in/d_W + d_a/d_W)
        #        = dL/d_out*d_a/d_h*dh/d_W
        # dL/d_b = dL/d_out*d_out/d_b ...
        dL_dh = grads
        dL_dh[self.h<=0] = 0

        dL_dW = self.inputs.T@dL_dh
        dL_db = dL_dh.sum(axis=0)

        return [(self.W, dL_dW), (self.b, dL_db), self.name]
```

Rješenje 2 NumPy:

```
import numpy as np

class ResBlock(Layer):
    def __init__(self, num_inputs, name):
        # 0.5 boda
        self.name = name
        self.FC = FC(num_inputs, num_inputs, name+".FC")
        self.relu = ReLU(num_inputs, name+".ReLU"

    def forward(self, inputs):
        # 0.5 boda
        h = self.FC.forward(inputs)
        a = self.relu(forward(h))
        return inputs + a

    def backward_inputs(self, grads):
        # 0.5 boda
        dL_dh = self.relu.backward_inputs(grads)
        dL_din_FC = self.FC.backward_inputs(dL_dh)
        return grads + dL_din_FC

    def backward_params(self, grads):
        # 0.5 boda
        dL_dh = self.relu.backward_inputs(grads)
        [dL_dW, dL_db, _] = self.FC.backward_params(dL_dh)
        return [dL_dW, dL_db, self.name]

# ...
# 3 boda - implementacija FC iz laboratorijske vjezbe
# 2 boda - implementacija ReLU iz laboratorijske vjezbe
# ...
```

Rješenje Pytorch:

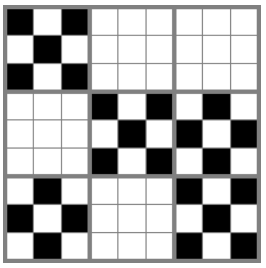
```
# 3 boda
import torch
import torch.nn as nn

class ResBlock(nn.Module):
    def __init__(self, num_in)
        super(ResBlock, self).__init__()
        self.FC = nn.Linear(num_in, num_in, bias=True)

    def forward(self, inputs)
        h = self.FC(inputs)
        a = torch.relu(h)
        return a + inputs
```

Rješenje TensorFlow:

```
import tensorflow as tf
import tensorflow.contrib.layers as layers
```

```
def build_res_layer(net):
    N, D = net.shape
    h = layers.fully_connected(net, D) # bias default true
    a = tf.nn.relu(h)
    return a + net
```

3. (10b) Vaš prijatelj X Æ A-12 je profesionalni e-sports igrač u igri "Kružić i križić" i treba pomoć. X Æ A-12 posjeduje bazu odigranih igara i želi izbrojati koliko puta je pobjedio igrajući kao **X** (križić). Završno stanje svake igre zabilježeno je binarnom slikom veličine 9x9 piksela. Slike u bazi su strukturirane na način da su znakovi uvijek pravilni i ucrtani unutar odgovarajućih zamišljenih ćelija veličine 3x3 piksela. Na slici desno vidimo primjer znaka **X** u ćeliji gore lijevo, a znaka **O** u ćeliji dolje lijevo. Crni pikseli na slici imaju vrijednost jednaku 1, a bijeli 0. Sivi rubovi zapravo ne postoje, dodani su samo zbog preglednosti. Vaši zadaci su sljedeći:



   (a) (5b) razmatramo model zadan nizom slojeva: konvolucijski sloj, potpuno povezani sloj, sloj globalnog sažimanja maksimumom, sigmoida; odrediti vrijednosti svih hiperparametara i parametara takve da model ispravno klasificira ulazne slike u dva razreda (pobjeda **X** i ostalo);

   (b) (2b) provesti jedan unaprijedni prolaz za ulaz zadan slikom i demonstrirati da Vaš model radi ispravno,

   (c) (3b) izvesti gradijente gubitka binarne unakrsne entropije po klasifikacijskim mjerama (ulazima sigmoide).

4. (3b) Razmatramo dvoslojni LSTM s dimenzijom skrivenog stanja 500 te dimenzijom ulaznih reprezentacija prvog sloja 300. Odredite ukupan broj parametara navedene mreže.

5. (7b) Razmatramo običnu povratnu neuronsku mrežu s prijenosnom funkcijom **logističke sigmoide** na povratnoj vezi i bez izlaznog sloja. Neka su poznati parametri povratne mreže:

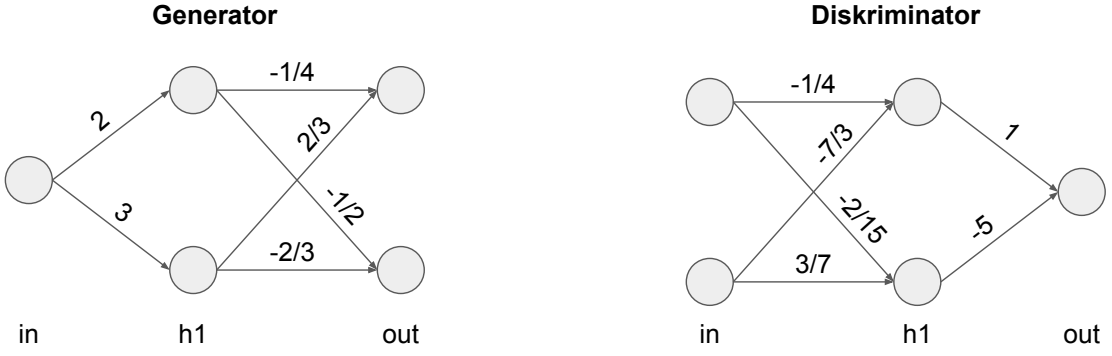
$$\boldsymbol{W}_{\text{hh}} = \begin{bmatrix} 2\ln(\sqrt{2}) & 0 \\ 0 & 2\ln(\sqrt{2}) \end{bmatrix} \qquad \boldsymbol{W}_{\text{xh}} = \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix} \qquad \boldsymbol{b}_{\text{h}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

   .

   Odredite skriveno stanje $\boldsymbol{h}^{(2)}$ nakon dva vremenska koraka ako je poznato da je početno skriveno stanje inicijalizirano na nul vektor, a ulazi u prva dva vremenska koraka su:

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} 2\ln(\sqrt{2}) \\ 2\ln(\sqrt{2}) \end{bmatrix} \qquad \boldsymbol{x}^{(2)} = \begin{bmatrix} 0 \\ 1.5\ln(\sqrt{2}) \end{bmatrix}$$

6. (10b) Zadan je osnovni GAN sljedeće arhitekture i težina.



   Za generator su aktivacijske funkcije na svim slojevima ReLU, a u diskriminatoru su $\mathrm{LeakyReLU}$ s negativnim nagibom vrijednosti 0.2, osim izlaznog neurona diskriminatora, čija je aktivacija sigmoidalna funkcija. U mreži su svi pomaci nepromijenjivi i jednaki 0.

   Pri treniranju se prvo korigiraju težine diskriminatora, a tek onda generatora.

(a) Opišite na koji način se trenira navedeni GAN.

(b) Navedite optimizacijski cilj treniranja.

(c) Za navedeni GAN izračunajte novo stanje težina sloja h_1 generatora i h_1 diskriminatora nakon jednog koraka učenja, uz stopu učenja od 1, s minigrupom veličine 3, ulaznim uzorcima šuma u generator vrijednosti -0.5, 0.5, 0 i skupom podataka od tri elementa: $[0, 0]$, $[1, 1]$ i $[2, 2]$.