26. ožujka 2021.

Zavod za elektroniku, mikroelektroniku,
računalne i inteligentne sustave

Duboko učenje

provjera znanja 1. laboratorijske vježbe

1. Zadana je funkcija $f(x) = x^4 - 32x^2 + 64$.

   (a) Odredite prvu derivaciju funkcije f.

   (b) Napišite kod u Numpy-u koji traži minimum zadane funkcije gradijentnim spustom.

   (c) Napišite istovjetni kod u Pytorch-u, ali korištenjem automatske diferencijacije.

   (d) Ovisi li rezultat izvođenja Vaše funkcije o odabiru početne točke? Objasnite! Postoji li inicijalizacija koja čak ni uz neograničen broj iteracija neće rezultirati pronalaskom minimuma?

2. Neka je zadan skup D-dimenzionalnih linearno nerazdvojivih podataka te odgovarajući vektor indeksa C razreda, uz D=10 i C=3.

   (a) Napišite jednadžbe unaprijednog prolaza diskriminativnog modela za nadzirano učenje na zadanim podatcima. Model treba imati jedan potpuno povezani skriveni sloj aktiviran zglobnicom te izlazni sloj koji ima probabilističku interpretaciju.

   (b) Odredite funkciju $f(H)$ koja računa broj parametara modela u ovisnosti o dimenzionalnosti skrivenog sloja H. Odredite H za model maksimalnog kapaciteta, ukoliko je najveći dozvoljeni broj parametara 49.

   (c) Implementirajte model u Pytorchu primjenom komponente `nn.Sequential` i prikladnog gubitka. Napišite kod koji ispisuje gradijente gubitka po težinama prvog sloja u slučajnom podatku.

3. Razmatramo sljedeći incijalizacijski kod za program strojnog učenja:

```
W1 = np.random.randn(D1,D)
b1 = np.random.randn(D1)
W2 = np.random.randn(C,D1)
b2 = np.random.randn(C)
X = my.get_data()      # shape: N,D
Y = my.get_labels()    # shape: N
```

   Unaprijedni prolaz modela prikazan je sljedećim kodom:

```
s1 = X @ W1.T + b1
h1 = np.maximum(0, s1)
s2 = h1 @ W2.T + b2 + s1
p = softmax(s2)
L = - np.mean(np.log(p[range(N),Y]))
```

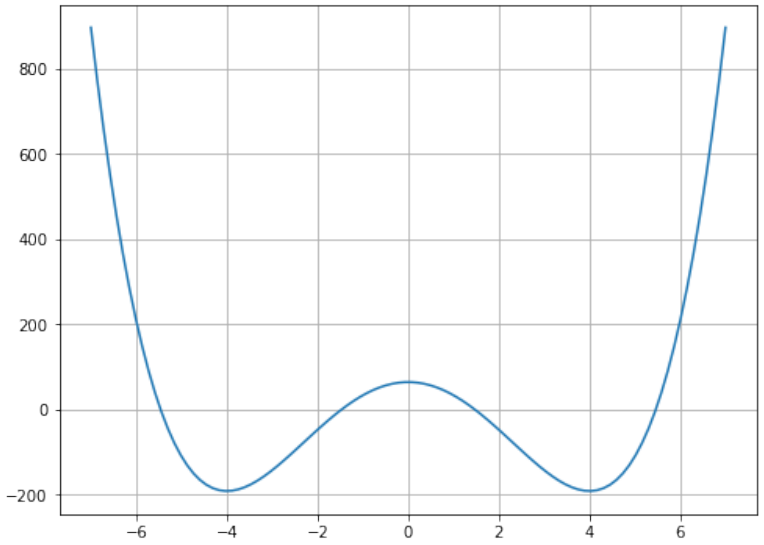
   Predložite kod u numpyju za unatražni prolaz.

1. (a) (2.5 boda) graf funkcije se nije tražio

$$f(x) = x^4 - 32x^2 + 64 \tag{1}$$

$$f'(x) = 4x^3 - 32 \cdot 2x \tag{2}$$

$$f'(x) = 4x^3 - 64x \tag{3}$$



Slika 1: Prikaz funkcije $f(x)$.

(b) Numpy (2.5 boda) :

```
def f(x):
    return x**4 - 32 * x ** 2 + 64

def d_f(x):
    return 4 * x ** 3 - 64 * x

x = -2
iters = 100
lr = 0.001
for i in range(iters):
    f_val = f(x)
    d_f_val = d_f(x)
    x = x - lr * d_f_val
    print(f"x␣=␣{np.round(x,3)}␣f(x)␣=␣{f_val}")
```

(c) Pytorch (2.5 boda):

```
lr = 0.001
x = torch.tensor(-2.0, requires_grad=True)
optim = torch.optim.SGD([x], lr=lr)
iters = 100
for i in range(iters):
    f_val = f(x)
    optim.zero_grad()
    f_val.backward()
    optim.step()
    print(f"x␣=␣{np.round(x.item(),3)}␣f(x)␣=␣{f_val}")
```

(d) (2.5 boda) Okej: Rezultat izvođenja funkcije ovisi o odabiru početne točke. Primjerice, uz neadekvatnu stopu učenja i inicijalizaciju velikim brojem, optimizacija može divergirati zbog prevelikih iznosa gradijenta.
Bolje: Funkcija u točki $x = 0$ ima lokalni maksimum u kojem je derivacija jednaka 0. Uz takvu inicijalizaciju optimizacija će sigurno biti neuspješna, jer je gradijent jednak nuli i zapravo nikada neće doći do ažuriranja.

Rješenje zadatka 2:

```
D=10
C=3

################################# 2.a (5 bodova)

s1 = X@W1.T + b1
s2 = max(0, s1)
s3 = s2 @ W2.T + b2
y_hat = softmax(s3)

################################# 2.b (5 bodova)

f(H, D, C) = D*H + H + H*C + C
f(H) = f(H, 10, 3) = 14H + 3

f(H) = 49 => H = floor((49-3)/14) = 3

################################# 2.c (BONUS)

import torch
import torch.nn as nn

D = 10
C = 3
H = 3

model1 = nn.Sequential(
        nn.Linear(D, H),
        nn.ReLU(),
        nn.Linear(H, C)
)
loss1 = nn.CrossEntropyLoss()

model2 = nn.Sequential(
        nn.Linear(D, H),
        nn.ReLU(),
        nn.Linear(H, C),
        nn.LogSoftmax(dim=1)
)

loss2 = nn.NLLLoss()

model, loss = model1, loss1


y = torch.ones(1)
x = torch.randn(1, D)
out = model(x)
loss(out, y).backward()

print(f"Weight␣grad:␣{model[0].weight.grad}")
print(f"Weight␣grad:␣{model[0].bias.grad}")

# Prediction
model.eval()
y_hat = F.softmax(model(x), dim=1).max(dim=1)[1]
```

Rješenje trećeg zadatka:

```
# 2 boda
dLds2 = (p - np.eye(C)[y])*1/N # dL_i/ds2_i  = p_i - Y^oh_i

# 2 boda
dLdW2 = dLds2.T @ h1           # dL_i/dW2_j: = dL_i/ds2_ij * ds2_ij/dW2_j:
dLdb2 = np.sum(dLds2, axis=0)  # dL_i/db2_j  = dL_i/ds2_ij * ds2_ij/db2_j:

# 2 boda
dLdh1 = dLds2 @ W2             # dL_i/dh1_i = dL_i/ds2_i * ds2_i/dh1_i
# 3 boda
dLds1 = dLdh1 * (s1>0) + dLds2 # dL_i/ds1_ij = dL_i/dh1_ij * (s1_ij>0)
                               #             + dL_i/ds2_ij

# 1 bod
dLdW1 = dLds1.T @ x
dLdb1 = np.sum(dLds1, axis=0);
```