Neprivalomos užduotys papildomam balui (II dalis, 0,5 b.)
Sprendimai, paaiškinimai

1. Kuo skiriasi stochastinis gradientinis nusileidimas (angl. *stochastic gradient descent*) nuo mini-rinkinių gradientinio nusileidimo (angl. *minibatch gradient descent*)? Kuriais atvejais naudotume vieną, kuriais kitą? (0,1 b.)

- ⋆ (klausimo esmė apie *batch size*; praktikoje nautotume kažką geriau nei *SGD*, pvz. *Adam*)
- stochastinis – cikle įvertina tik vieno duomens gradientą ir su juo atnaujina parametrus; mini-rinkinių – cikle įvertina kelių duomenų gradientą ir su juo atnaujina parametrus;
- dažniausiai norime stabilaus mokymo, tad stengtumėmės naudoti pilno rinkinio arba mini-rinkinių gradientinį nusileidimą su dideliu rinkinio dydžiu (angl. *batch size*);
- naudotume stochastinį jei norime greitesnio mokymo, tačiau mažiau stabilaus; arba kiekvienas individualus duomo užima pakankamai daug atminties;
- ⋆ stochastinis gali padėti išvengti lokalių minimumų paieškos erdvėje.

2. Kaip reiktų pakeisti stochastinio gradientinio nusileidimo algoritmą, jei kažkuri gradiento koordinatė būtų pastoviai didesnė nei kitos? (Pvz. gradiento $\nabla_\theta \mathcal{L} = \begin{bmatrix} 0.63 & -0.47 & 129.58 & 1.24 \end{bmatrix}$ trečioji koordinatė ženkliai didesnė nei kitos.) (0,1 b.)

- ⋆ (nėra vieno teisingo atsakymo, pateikiu kelis:)
- galime mokymo metu „apkirpti“ gradientus (angl. *gradient clipping*);
- užtikrinti, kad įvesties požymiai yra normalizuoti (angl. *batch normalization*);
- kiekvienam parametrui suteikti individualų žingsnio dydį (angl. *individual learning rates*);
- naudoti kokį nors mokymosi algoritmą, kuris adaptuotųsi prie gradientų dydžių (angl. *adaptive learning rate*), pvz. *AdaGrad*, *Adam*, *RMSProp*.

3. Kaip reiktų pakeisti klasikinę tekstinę paiešką, kad ji galėtų atsakyti užklausą apytiksliai? Pateikite pavyzdį. Galite remtis *D2L* knyga (*Dive into Deep Learning, Zhang et al. 2023. Chapter 11. Attention Mechanisms and Transformers*). (0,1 b.)

- ⋆ (nėra vieno teisingo atsakymo, pateikiu vieną iš galimų:)
- padalinti tekstą ir užklausą į dalis (angl. *tokens*), pvz. žodžius, kuriuos paverčiame į įterpinius (angl. *embeddings*) su kokiu nors esamu ištreniruotu (angl. *pre-trained*) modeliu;
- suskaičiuoti panašumo arba dėmesio (angl. *attention*) funkciją tarp užklausos ir teksto;
- su slenkstine reikšme išrinkti geriausius rezultatus.

Pavyzdys:
Tekstas: `Rytais geriu juodąją arbatą, vakarais žaliąją arbatą.`
Užklausa: `žalioji arbata`
Įterpiniai (paprastumo dėlei įterpinių dydis lygus 2):

`rytais`	`geriu`	`juodąją`	`arbatą`	`vakarais`	`žaliąją`	`žalioji`	`arbata`	…
$(0.2, 0.3)$	$(-0.3, 0.2)$	$(0.2, -1.1)$	$(-0.8, 0.4)$	$(0.6, 0.2)$	$(0.3, -1.0)$	$(0.4, -1.0)$	$(-0.9, 0.4)$	…

Paverčiame tekstą ir užklausą į įterpinių vektorius:
Tekstas: $[(0.2, 0.3), (-0.3, 0.2), (0.2, -1.1), (-0.8, 0.4), (0.6, 0.2), (0.3, -1.0), (-0.8, 0.4)]$;
Užklausa: $(0.4, -1.0), (-0.9, 0.4)$.
Pasirenkame panašumo funkciją, pvz. vektorinę sandaugą, ir suskaičiuojame ją tarp užklausos ir teksto kiekvienoje pozicijoje: $[0.13, -0.94, 2.06, -1.18, -0.63, 2.0]$.
Galime išrinkti didžiausią reikšmę kad išgautume viena atsakymą, arba pritaikyti *sigmoid* funkciją ir slenkstį (pvz. 0.6), kad išgautume daug atsakymų: $[0.53, 0.28, 0.89, 0.24, 0.35, 0.88]$.
Rezultatas – didžiausias panašumas 3 ir 6 pozicijoje, t.y., `juodąją arbatą` ir `žaliąją arbatą`.

Jei įterpinių matrica būtų kitokia, gautume kitokį rezultatą. Remiamės prielaida, kad ši matrica yra mums duota, ir tinkamai užkoduoja žodžių reikšmes.

4. Spręsdami vaizdų lokalizavimo uždavinius dažnai modelio išvestyje gauname daugelį persidengiančių stačiakampių. Dažnai naudojame nemaksimalaus apjungimo (angl. *non-maximum suppression*) algoritmą, kad pašalintume didžiąją dalį stačiakampių. Šis algoritmas yra godus (angl. *greedy*)[1]. Ar gali atsitikti taip, kad su šiuo algoritmu pašalinsime naudingus stačiakampius? Kaip galėtume pakeisti šį algoritmą, kad ne taip griežtai išmestume stačiakampius, ir išgautume geresnius spėjimus? Galite remtis *Soft-NMS* šaltiniu (*Soft-NMS – Improving Object Detection With One Line of Code*, Bodla et al., 2017.). (0,1 b.)

- taip, *NMS* gali pašalinti naudingus stačiakampius, ypač kai yra keli šalia esantys (persidengiantys) objektai;
- galime ne taip griežtai išmesti stačiakampius su *Soft-NMS* – šis metodas sumažina persidengiančių stačiakampių įverčius, tačiau jų neišmeta;
- įverčius sumažiname pagal atstumą iki „geriausio“ stačiakampio, išsaugodami artimiausių stačiakampių informaciją;
- ⋆ yra daugiau galimybių pasirinkti tinkamus stačiakampius, pvz. iš kelių persidengiančių stačiakampių gauname vieną pasvertinio vidurkio (angl. *weighted average*), pagal spėjimo įvertį, būdu.

5. Kartais *ReLU* aktyvacijos funkcija gali „pradingti“ (angl. *dead ReLU problem*) – t.y., su bet kokia įvestimi neurono išvestis lygi nuliui, o tai reiškia, kad jo išvestinė taip pat tampa lygi nuliui. Taip nutikus modelis lėčiau mokosi ir gali neišmokti kai kurių požymių. Kaip reiktų spręsti šią problemą? (0,1 b.)

- naudoti *Leaky ReLU*, *ELU*, arba *Parametric ReLU* aktyvacijos funkciją, tuomet išvestinė nebus lygi nuliui net jei įvestis neigiama;
- ⋆ (galimi ir kiti atsakymo variantai)

[1] Godus algoritmas (angl. *greedy algorithm*) – algoritmas, kuris kiekviename etape priima lokaliai optimalius sprendimus. Kai kuriuose uždaviniuose lokaliai optimalūs sprendimai priveda prie globaliai optimalaus sprendimo (pvz. trumpiausio kelio grafe radimas naudojant *Dijkstra* algoritmą).