Търсене и извличане на информация. Приложение на дълбоко машинно обучение

Зимен семестър 2025/2026

Домашно задание №2

20 декември 2025 г.

Общ преглед

В това задание ще имплементираме word2vec skip-gram noise contrastive estimation with general multiplicative measure модел за научаване на влагане на думи. Обучението на влаганията U, V и параметъра W от корпус от наблюдения $\mathbf{X}$ в този модел се свежда до минимизацията на функцията на загубата:

$$J_{\mathbf{X}}(U,V,W) \;=\; -\frac{1}{|\mathbf{X}|}\sum_{(w,\mathbf{c})\in\mathbf{X}}\sum_{c_i\in\mathbf{c}}\left(\log\sigma(\mathbf{v}_{c_i}^T W\mathbf{u}_w - q_{c_i}) + \sum_{j=1}^{n}\log\sigma(q_{\bar{c}_{i,j}} - \mathbf{v}_{\bar{c}_{i,j}}^T W\mathbf{u}_w)\right),$$

където ако $M \in \mathbb{N}$ е размерността на влагането, а L е множеството от думи в нашия речник, то:

- $U, V \in \mathbb{R}^{|L|\times M}$ са влаганията съответно на целевата дума и на контекстните думи
- $\mathrm{W} \in \mathbb{R}^{M\times M}$ е квадратична форма – параметър на модела
- w е индексът на целевата дума
- $\mathbf{c}$ е списък от индексите на контекстните думи c_i
- $\bar{c}_{i,j}$ са индексите на n различни случайно избрани думи от нашия речник, които са различни от c_i.

Влагането на центъра w означаваме с $\mathbf{u}_w = U_{w,\bullet} \in \mathbb{R}^M$, влагането на контекстна дума c означаваме с $\mathbf{v}_c = V_{c,\bullet} \in \mathbb{R}^M$, а $q_c = \log nq(c)$, където $q(\cdot)$ е разпределението на "шума". $\sigma(z) = \frac{1}{1+e^{-z}}$ е логистичната функция.

Дефинираме функцията на загубата в точката $w, c, \bar{c}_1, \ldots, \bar{c}_n$ като:

$$J_{w,c,\bar{c}_1,\ldots,\bar{c}_n}(U,V,W) = -\left(\log\sigma\left(\mathbf{v}_c^T W \mathbf{u}_w - q_c\right) + \sum_{j=1}^{n} \log\sigma(q_{\bar{c}_j} - \mathbf{v}_{\bar{c}_j}^T W \mathbf{u}_w)\right),$$

Свойства на логистичната функция

Тук ще припомним някои свойства на логистичната функция, които може да ви послужат във вашето решение.

$$\begin{aligned} 1-\sigma(z) &= \frac{e^{-z}}{1+e^{-z}} \\ \sigma(-z) &= 1-\sigma(z) \\ \sigma'(z) &= \sigma(z)(1-\sigma(z)) \\ (\log\sigma(z))' &= 1-\sigma(z) \end{aligned}$$

Задача 1 (1 точки)

Изразете частните производни

1. $\frac{\partial J_{w,c,\bar{c}_1,\ldots,\bar{c}_n}(U,V,W)}{\partial \mathbf{u}_w} =$
2. $\frac{\partial J_{w,c,\bar{c}_1,\ldots,\bar{c}_n}(U,V,W)}{\partial \mathbf{v}_c} =$
3. $\frac{\partial J_{w,c,\bar{c}_1,\ldots,\bar{c}_n}(U,V,W)}{\partial \mathbf{v}_{\bar{c}_j}} =$
4. $\frac{\partial J_{w,c,\bar{c}_1,\ldots,\bar{c}_n}(U,V,W)}{\partial W} =$

като използвате единствено параметрите $\mathbf{u}_w$, $\mathbf{v}_c$, $\mathbf{v}_{\bar{c}_j}$, q_c, $q_{\bar{c}_j}$ и W. Търсим решение, което се изразява просто – всеки от получените изрази би следвало да се събира на един ред.

Важно: Резултатни изрази, които съдържат други параметри извън гореуказаните, дори да са коректни, няма да бъдат приети.

Задача 2 (2 точки)

Нека $\tilde{V} \in \mathbb{R}^{(n+1)\times M}$ е подматрица на V, която е съставена от редовете, съответстващи на контекстните думи $c, \bar{c}_1, \ldots, \bar{c}_n$, векторът $\bar{q} \in \mathbb{R}^{n+1}$ е със стойности $q_c, q_{\bar{c}_1}, q_{\bar{c}_2}, \ldots, q_{\bar{c}_n}$, а векторът $\bar{\delta}_c \in \mathbb{R}^{n+1}$ е $n+1$ мерен вектор с единица на първа позиция и нули на всички останали. Т.е.

$$\tilde{V} = \begin{bmatrix} \mathbf{v}_c \\ \mathbf{v}_{\bar{c}_1} \\ \mathbf{v}_{\bar{c}_2} \\ \vdots \\ \mathbf{v}_{\bar{c}_n} \end{bmatrix}, \bar{q} = \begin{bmatrix} q_c \\ q_{\bar{c}_1} \\ q_{\bar{c}_2} \\ \vdots \\ q_{\bar{c}_n} \end{bmatrix}, \bar{\delta}_c = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Изразете частните производни

1. $\frac{\partial J_{w,c,\bar{c}_1,\ldots,\bar{c}_n}(U,V,W)}{\partial \mathbf{u}_w} =$
2. $\frac{\partial J_{w,c,\bar{c}_1,\ldots,\bar{c}_n}(U,V,W)}{\partial \tilde{V}} =$
3. $\frac{\partial J_{w,c,\bar{c}_1,\ldots,\bar{c}_n}(U,V,W)}{\partial W} =$

като използвате единствено параметрите $\mathbf{u}_w$, $\tilde{V}$, W, $\bar{q}$ и $\bar{\delta}_c$. Търсим решение, което се изразява просто – да се изразява с матрични и векторни операции и да не съдържа експлицитни сумирания. Всеки от получените изрази би следвало да се събира на един ред.
Важно: Резултатни изрази, които съдържат други параметри извън гореуказаните, дори да са коректни, няма да бъдат приети.

Задача 3 (1 точки)

Тук ще започнем с имплементационната работа. За семплирането[1] на негативните примери е необходимо да избираме негативната контекстна дума c с вероятност за шум:

$$q[c[= \Pr_{0.75}[c] = \frac{\text{round}(\#(c)^{0.75})}{\sum_{w \in L} \text{round}(\#(w)^{0.75})},$$

където $\#(w)$ е броят на срещанията на думата w. Семплирането се извършва във функцията `sampleContext` във файла `sampling.py`, където

[1] Тази функционалност може да се постигне чрез параметъра `p` на функцията `numpy.random.choice`, но използването ѝ е значително по-бавно от нашата имплементация

се избират n случайни елемента от списъка `seq`. Елементите на списъка `seq` следва да бъдат номера на термове от речника. За да се получи семплиране с горната вероятност, броят на срещанията на номера на даден терм в списъка `seq` трябва да бъде пропорционален на неговата вероятност. В нашата реализация този брой следва да бъде $\#(w)^{0.75}$ закръглено до най-близкото цяло число. Задачата е да се попълни функцията `createSamplingSequence` във файла `sampling.py`, която по даден списък `freq` от честотите на термовете, генерира списъка `seq`.

Освен функцията за семплиране ще ни бъде необходимо предварително да изчислим стойностите q_c за всяка дума c, зададени с формулата $q_c = \log(nq[c])$. За целта следва да попълните тялото на функцията `noiseDistribution` във файла `sampling.py`.

След завършване е желателно да тествате вашата имплементация чрез командата:

```
python test.py 3
```

Задача 4 (2 точки)

Тук ще имплементираме изчислението на градиентите и функцията на загубата. За целта е необходимо да се попълни функцията `lossAndGradient` във файла `grads.py`. В тази функция следва по зададени вектор $\mathbf{u}_w$, в програмата означен с `u_w`, матрица $\tilde{V}$, в програмата означена с `Vt`, матрица W и вектора $\mathbf{q}$ да се сметне загубата в точката и градиентите по $\mathbf{u}_w$, $\tilde{V}$ и W. Очаква се да имплементирате формулите, които сте получили в Задача 2, като използвате ефективни numpy векторни операции.
След завършване е желателно да тествате вашата имплементация чрез командата:

```
python test.py 4
```

Задача 5 (2 точки)

Имплементацията от Задача 4 изчислява градиента за всяка точка от данните поотделно. Това не позволява оптималното използване на паралелната архитектура на съвременните компютърни системи. При партидния стохастичен градиент се налага да изчисляваме градиентите за цяла партида от наблюдения. Поради това ще трябва да имплементирате ефективен партиден вариант на функцията за изчисляване на градиента. Попълнете функцията `lossAndGradientBatched` във файла `grads.py`.

Тук заедно с матрицата W едновременно се подават векторите $\mathbf{u}_w$, матриците $\tilde{V}$, както и векторите $\mathbf{q}$ за S наблюдения. Очаква се вместо да акумулирате резултатите за отделните наблюдения, да използвате тензорни операции, чрез които наведнъж да получите резултатите за цялата партида. Очаква се по този начин да получите над 2-пъти по-бързо изпълнение.

ЗАБЕЛЕЖКА: Градиентите и загубата за партидата следва да са нормализирани спрямо броя на наблюденията. След завършване е желателно да тествате вашата имплементация чрез командата:

```
python test.py 5
```

Задача 6 (2 точки)

Време за експерименти.
Първо, довършете имплементацията на функцията `stochasticGradientDescend` във файла `w2v_sgd.py`. Трябва да допълните функцията с команди за презаписването на параметрите U, V и W като за всяко от наблюденията от партидата се извади α умножено по съответният градиент. След завършване е желателно да тествате вашата имплементация чрез:

```
python test.py 6
```

Сега може да пристъпите към пълното обучение на модела. Ако сте се справили със Задача 4, може да обучите влагане чрез изпълнението на командата:

```
python run.py
```

Изпълнението би следвало да отнеме около 1 – 2 часа. Като резултат следва да получите грешка около и под 2.4. Също така, в текущата директория би следвало да се запишат файловете `w2v-U.npy`, `w2v-V.npy`, `w2v-W.npy` и `embeddings.png`.

В случай че успешно сте получили влаганията по гореописания начин, не е необходимо да правите нещо повече. Ако имате проблем или не сте успели да имплементирате Задача 5, но сте се справили със Задача 4, може да обучите влагане чрез изпълнението на командата:

```
python run.py cumulative
```

В този случай изпълнението би следвало да отнеме около 3 – 4 часа и да се получат същите резултати от изпълнението.

Инструкция за предаване на домашна работа

Напомняме, че успешното минаване на приложените тестове не е достатъчно решението ви да бъде прието. За приемането му ще бъдат приложени по-задълбочени тестове.

Изисква се в Moodle да бъде предаден архив FNXXX.zip (където XXX е вашият факултетен номер), който съдържа:

1. Файловете `grads.py`, `w2v_sgd.py`, `sampling.py` съдържащи нанесените от вас промени
2. Решения на Задачи 1 и 2. Решенията може да са под форма на сканирани/снимани записки на хартия или pdf.
3. Файловете `w2v-U.npy`, `w2v-V.npy`, `w2v-W.npy` и `embeddings.png` получени след изпълнението на Задача 6