Основи машинског учења, јесен 2021. домаћи задатак №7

Рок: недеља, 9. јануар у 23:59 на Moodle-у.

Упутства: (1) Ова питања захтевају размишљање, не и дуге одговоре. Будите што сажетији. (2) Уколико има било каквих нејасноћа, питајте предметног наставника или сарадника. (3) Студенти могу радити и послати решења самостално или у паровима. У случају заједничког рада, имена и презимена оба студента морају бити назначена у Gradescope-у и није дозвољено радити са истим колегом више од једном. (4) За програмерске задатке, коришћење напредних библиотека за машинско учење попут scikit-learn није дозвољено. (5) Кашњење приликом слања односно свака пошиљка након рока носи негативне поене.

Сви студенти морају послати електронску PDF верзију својих решења. Препоручено је куцање одговора у LaTeX-у које са собом носи 10 додатних поена. Сви студенти такође морају на Moodle-у послати и zip датотеку која садржи изворни код, а коју би требало направити користећи `make_zip.py` скрипту. Обавезно (1) користити само стандардне библиотеке или оне које су већ учитане у шаблонима и (2) осигурати да се програми извршавају без грешки. Послати изворни код може бити покретан од стране аутоматског оцењивача над унапред недоступним скупом података за тестирање, али и коришћен за верификацију излаза који су дати у извештају.

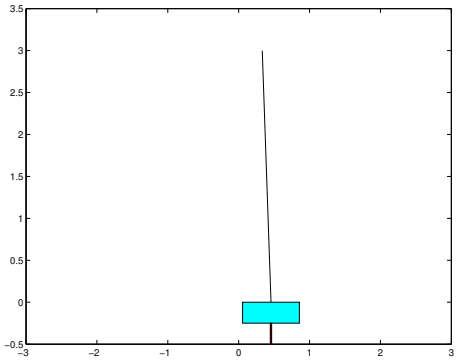
Кодекс академске честитости: Иако студенти могу радити у паровима, није дозвољена сарадња на изради домаћих задатака у ширим групама. Изричито је забрањено било какво дељење одговора. Такође, копирање решења са интернета није дозвољено. Свако супротно поступање сматра се тешком повредом академске честитости и биће најстроже кажњено.

1. **[90 поена] Учење подстицајем: обрнуто клатно**

У овом задатку биће примењено учење подстицајем да се аутоматски одреди политика за релативно сложен управљачки проблем без залажења у појединости динамике система којим се управља.

Задатак који ће бити размотрен је такозвани проблем обрнутог клатна или балансирања стуба.[1]

Размотрити приказану слику. Танак стуб је преко шарке повезан са колицима која се могу кретати латерално по глаткој површини стола. Каже се да је контролер отказао ако угао стуба одступа више од одређене границе у односу на вертикални положај, то јест ако је стуб пао или ако положај колица изађе ван оквира, то јест ако падну са ивице стола. Циљ је развити контролер који балансира стубом унутар ових ограничења тако што ће колица на одговарајући начин убрзавати лево и десно.



За овај задатак направљен је једноставан симулатор. Симулација се одвија у дискретним временским корацима. Стање колица и стуба у било ком тренутку у потпуности одређују четири параметра: положај колица x, брзина колица $\dot{x}$, угао стуба θ мерен као његово одступање од вертикалног положаја, као и угаона брзина стуба $\dot{\theta}$. Пошто је једноставније разматрати учење подстицајем у дискретном простору стања, простор стања је апроксимиран дискретизацијом која пресликава вектор стања $(x, \dot{x}, \theta, \dot{\theta})$ у број од 0 до `NUM_STATES-1`. Стога ће алгоритам учења подстицајем радити само над овом дискретном представом стања.

У сваком тренутку у времену, односно временском кораку, контролер мора изабрати једну од две акције - гурнути (убрзати) колица удесно или гурнути (убрзати) колица улево. (Да би се задатак у потпуности поједноставио, не постоји *не ради ништа* акција.) Претходне две акције су у изворном коду представљене са 0 и 1, респективно. Када се изабере једна од две акције, симулатор ажурира параметре стања у складу са динамиком система и враћа ново дискретизовано стање.

Биће усвојена претпоставка да је награда $R(s)$ функција само тренутног стања. Када угао стуба пређе одређену границу или када колица оду предалеко, додељује се негативна награда, а систем се поново иницијализује у случајно почетно стање. Иначе, у сваком другом тренутку, награда је нула. Програм мора научити да балансира стуб користећи само опажане односно посматране прелазе стања и награде.

Датотеке за овај задатак се налазе у `src/cartpole/` директоријуму. Већина кода је већ написана, а потребно је само допунити `cartpole.py` датотеку на назначеним местима. Ова

[1] Динамика је преузета са `https://all.cs.umass.edu/rlr/domains.html`

датотека може се покренути како би се добио графички приказ и како би се на крају исцртале криве учења. За више детаља о раду симулатора прочитати коментаре у заглављу датотеке.

Да би се решио задатак обрнутог клатна, неопходно је најпре проценити модел (односно, вероватноће прелаза и награде) за Марковљев поступак одлучивања (МПО) који је у основи проблема, затим решити Белманове једначине за тако процењен МПО да би се добила функција вредности, и најзад, деловати похлепно у односу на ту функцију вредности.

Укратко, биће одржаван тренутни модел МПО и тренутна процена функције вредности. У почетном тренутку, свако стање има процењену награду нула, а процењене вероватноће прелаза су униформне (подједнако је вероватно да се заврши у било ком другом стању).

Током симулације, у сваком временском тренутку морају се изабрати акције у складу са неком тренутном политиком. Како програм буде напредовао у извршавању радњи, прикупљаће запажања о преласцима и наградама које може искористити за бољу процену МПО модела. Пошто је крајње неефикасно ажурирати цео процењени МПО након сваког посматрања, опажања о променама стања и наградама биће чувана у сваком кораку, а модел и функција вредности, односно политика ће бити ажурирани само периодично. Стога је неопходно чувати укупан број опажених прелазака из стања s_i у стање s_j користећи акцију a , а слично томе и за награде. Треба имати у виду да су награде у било ком стању детерминистичке, али промене стања услед дискретизације простора стања нису (неколико блиских, али различитих конфигурација се могу пресликати у једно те исто дискретизовано стање).

Сваки пут када дође до отказа (на пример, ако стуб падне или колица оду ван стола), требало би поново да се процене вероватноће прелаза и награде као просек посматраних вредности (ако их уопште има). Програм тада мора користити итерацију по вредности да реши Белманове једначине на процењеном МПО како би добио функцију вредности и нову оптималну политику за нови модел. За итерацију по вредности треба користити критеријум конвергенције који проверава да ли максимална апсолутна промена функције вредности у некој итерацији премашује одређену унапред задату толеранцију.

Коначно, претпоставити да је целокупан поступак учења конвергирао када сваки од неколико узастопних покушаја (дефинисаних параметром `NO_LEARNING_THRESHOLD`) решавања Белманових једначина конвергира у првој итерацији. Интуитивно, ово указује да је процењени модел престао да се значајно мења.

Нацрт изворног кода за овај задатак се већ налази у `cartpole.py` и потребно је само дописати делове кода у назначеним местима унутар датотеке. Постоји и неколико додатних појединости (критеријуми конвергенције и слично) који су такође додатно појашњени унутар самог изворног кода. Користити фактор попуста $\gamma = 0.995$.

Имплементирати алгоритам учења подстицајем као што је наведено и покренути га.

- Колико покушаја (односно колико пута су стуб или колица пали) је било потребно пре него што је алгоритам конвергирао? Смерница: у тачном решењу, на графику који представља логаритам усредњеног броја корака до отказа, црвена линија треба да почне да се заравњује након шездесетак итерација.
- Нацртати криву учења која приказује број временских корака за коју је стуб балансиран за сваку симулацију. Укључити график у извештај.
- Променити `np.random.seed` на вредности вредностима 1, 2 и 3 и поново покренути програм. Која су запажања? Шта ово говори о самом алгоритму?