Organizační úvod

Poznámka (Organizační úvod)
Zápočet není nutný na zkoušku. Zkouška bude mít dvě části: první bude problém, který máme nějak vyřešit pomocí probraných algoritmů, druhá pak 2 otázky z teorie. Streemování nebude, budou pravděpodobně přednášky z minulého roku (asi na Teams).

1 Úvod

Definice 1.1 (AI)

Umělou inteligenci máme hlavně dvou druhů, symbolickou (pracuje nad symbolicky (formálním jazykem – např. matematickou logikou) popsaným světem, řeší například plánování a reprezentaci znalostí) a výpočetní (pracuje přímo s „reálným světem“ = daty, například ML (NN, DL, k-means, stromy), evoluční algoritmy a další přírodou inspirované algoritmy).

Symbolická AI může být třeba (klasicky taková AI má počáteční stav a akci se jménem, předpokladem, přidáním efektů a ubráním efektů):

```
on(B, A)
on(table, C)
clear(A), clear(C)

pick(X)
predpoklad: clear(X), on(A, X)
+efekty: holding(X), clear(A)
-efekty: clear(X), on(A, X)
```

Poznámka
Dále jsme si povídali o jednoduchých základech NN.

Definice 1.2 (Evoluční algoritmy)

Řeší nějakou optimalizaci (hledání minima/maxima). Funguje tak, že máme nějakou populaci bodů, nějakým způsobem je křížíme + mutujeme a udržujeme velikost populace odebráním horších členů (hodnocení členů se nazývá fitness).

Definice 1.3 (Učení s učitelem, učení bez učitele, zpětnovazebné učení)

Učení s učitelem je, že máme zadaná nějaká data i s výsledky (dělí se na klasifikaci = předpověď kategorie a regresi = předpověď „spojitého“ čísla).

Učení bez učitele není, „že bych vás tu teď opustil a učte se sami“, ale že nejsou dané správné odpovědi.

Zpětnovazebné učení je, když cílem agenta je maximalizovat nějakou zpětnou vazbu z prostředí (většinou vyjadřovanou jako číslo, kladné je „odměna“, záporná „trest“).

2 Zpětnovazebné učení

Definice 2.1 (Mountain Car („autíčko v ďolíčku“))

Auto je v 2D údolí a nemá výkon na to, aby vyjelo nahoru přímo. Cílem je samozřejmě dostat se nahoru (pomocí akcí dopředu, dozadu, neutral). Odměna je -1 za každý krok v prostředí (před dojetím do cíle).

(Existuje i spojitější verze, kde akce – reálné číslo mezi -1 a 1 – udává zrychlení)

Definice 2.2 (Zpětnovazebné učení)

Ve zpětnovazebném učení máme nějakého agenta, který provádí akce v prostředí a dostává informaci o stavu a odměně (ta se často dá spočítat ze stavu, ale pro jednoduchost rozlišujeme stav a odměnu). Formálně: Agent dostává stav s_t a provede akci a_t.

Definice 2.3

Podobně jako v AIUvod máme spojité a diskrétní a deterministické a nedeterministické prostředí.

2.1 Markovské rozhodovací procesy

Definice 2.4 (Markovský rozhodovací proces)

Markovský rozhodovací proces je čtveřice (S, A, P, R), kde S je množina stavů, A je množina akcí (občas to bývá funkce ze stavů – v každém stavu lze provést různé akce), $P_a(s, s')$ je přechodová funkce – pravděpodobnost, že aplikací $a \in A$ v $s \in S$ přejde prostředí do $s' \in S$, $R_a(s, s')$ je odměna, kterou dostane agent při přechodu z $s \in S$ do $s' \in S$ pomocí $a \in A$. P splňuje markovskou podmínku, tj. nezávisí na historii, závisí opravdu jen na a, s, s'.

Definice 2.5 (Strategie (policy))

Chování agenta popisujeme pomocí strategie $\pi(s, a)$, což je pro každé s pravděpodobnostní distribuce akcí.

Definice 2.6 (Diskontovaná odměna)

Cílem je maximalizovat odměnu (přes různé volby π): $\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$, kde $a_t = \pi(s_t)$ je akce provedená agentem v kroku t a $\gamma < 1$ je diskontní faktor, který zajišťuje, že suma konverguje, nastavuje, jak moc je důležité získat odměny co nejdřív, …

$V^\pi(s) = E[R] = E[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s]$, kde R je diskontovaná odměna, r_t je odměna získaná v čase t. $Q^\pi(s, a)$ je očekávaná odměna, kterou dostaneme, když ve stavu s uděláme

akci a a budeme pokračovat dál strategií π, říkáme tomu hodnota akce a ve stavu s.

Cílem agenta je tedy najít optimální strategii π^* takovou, že maximalizuje V^π. Hodnotu stavů a akcí pro optimální strategii budeme značit s $*$ místo π.

Definice 2.7 (ε-greedy strategie)

S pravděpodobností $1-\varepsilon$ vybere nejlepší akci (podle známých ohodnocení) a s pravděpodobností ε zvolí náhodnou akci.

Definice 2.8 (Monte-Carlo metody)

Pro výpočet $V^\pi(s)$ odsimulujeme n-krát budoucnost a zprůměrujeme odměny.

Definice 2.9 (Temporal-difference metody)

TD metody upravují ohodnocení stavů $V(s) \leftarrow V(s) - \alpha(r + \gamma V(s') - V(s))$.

Definice 2.10 (Q-učení)

Q-učení funguje podobně jako temporal-difference metody, jen upravují Q místo V, a to pomocí toho, že $V(s) = \max_a Q(s,a)$. Tradičně je Q reprezentováno jako matice, která je na začátku nulová a následně se upravuje podle:

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot \max_a Q(s_{t+1}, a) \right).$$

Poznámka
Lze si všimnout, že tu není potřeba znát pravděpodobnosti.

Definice 2.11 (SARSA)

Skoro jako Q-učení, jen místo maxima se kouká o krok dále a používá aktuální strategii agenta.

3 Evoluční algoritmy

Definice 3.1 (OneMAX)

OneMAX je problém na prostoru $\{0,1\}^N$, kde chceme dosáhnout nějaký (neznámý) pattern, např. samé jedničky. Fitness funkce bude počet správných prvků, např. počet jedniček.

Definice 3.2 (Evoluční algoritmus násvosloví)

Jeden „krok“ se jmenuje generace, jedinci použití na výrobu jiného se nazývají jeho rodiče, on se nazývá potomek.

Definice 3.3 (Genetický algoritmus)

Genetický algoritmus je evoluční algoritmus, který funguje na problému kódovaném jedničkami a nulami.

Máme nějakou populaci jedinců. Na nich provedeme selekci (pomocí fitness), pak křížení (nejčastěji jednobodové, tedy že od jednoho bodu prohodíme posloupnost jedinců, ale může být i uniformní, tedy že vyberu u každého bodu náhodně). Následuje mutace, kde náhodně přehodíme bity jedinců a pak začínáme od znova.

Selekce může být ruletová (pravděpodobnost výběru jedince je fitness jedince dělená součtem fitness všech, předpokladem je fitness ≥ 0).

Definice 3.4 (Elitismus)

Občas se při náhodné selekci nenáhodně vyberou do dalšího kroku dva nejlepší a nevyberou dva nejhorší jedinci.

Dále se probírali různé genetické operátory na jedincích složených s čísel $0, \ldots, k-1$, na reálných číslech a diferenční evoluci.

Definice 3.5 (Genetické programování)

Nelze očekávat, že g. p. vygeneruje složité programy. Lze ho použít např.

Definice 3.6 (Lineární genetické programování)

Vezme se nějaký triviální jazyk (stylu assembleru) a jedinci jsou posloupnost instrukcí v tomto jazyce. Pak můžeme dělat mutace: přidání, odebrání, změnění instrukce, změnění parametrů, … Jednobodové křížení pořád docela funguje.

Těžší je to s fitness. Program, který vyhodí chybu bude mít pravděpodobně fitness 0. Místo počítání času se často počítají instrukce, které se provedou.

Definice 3.7 (Kartézské genetické programování)

Kartézské genetické programování používá „síť“ uzlů, kde v uzlu volíme funkci a pak volíme, které výstupy jdou jako vstup do kterého uzlu (vždy jen zleva = od vstupu programu doprava = k výstupu programu).

Definice 3.8 (Gramatická evoluce)

Reprezentuje jedince jako posloupnost voleb, za co přepsat dotyčný neterminál..

Definice 3.9 (Stromové genetické programování)

Jedince reprezentujeme stromem.

4 Neuronové sítě

Přeskočeno (byl jsem na Strojovém učení v Pythonu a chodím na Deeplearning).

Definice 4.1 (Radial basis ? (RBF))

Do neuronových sítí nemusíme dávat přímo souřadnice, ale můžeme jim dávat na vstup i třeba vzdálenosti ($e^{-\beta||\mathbf{x}-\mathbf{c}||}$, kde $\mathbf{c}$ jsou středy, β parametry, jak moc klesá hodnota s poloměrem) od nějakých bodů. Ty najdeme nejčastěji pomocí k-means.

Dále se probíraly konvoluční neuronové sítě.

5 Neuroevoluce

Definice 5.1 (Evoluce vah v NN)

Podíváme se na neuronové sítě jako na vektory reálných čísel a použijeme genetický algoritmus.

Definice 5.2 (Evoluce topologie v NN)

(Např. neural architecture search.)

TODO!!!