Nekonvenční algoritmy a výpočty (NAVY)

Michael Ceplý - CEP0037

1. Perceptron

1.1 Úvod a cíl

Cílem této úlohy je vytvořit jednoduchý perceptron, který na základě náhodně generovaných 2D bodů rozhodne, zda leží pod nebo nad přímkou definovanou funkcí **y = 3x + 2**.
Perceptron se učí podle chyby a upravuje své váhy a bias, aby správně klasifikoval body.

1.2 Metodologie

Model:

- Jeden perceptron se dvěma vstupy.

Aktivační funkce:

- Signum – výstup je určen znaménkem lineární kombinace vstupů a biasu.

Učení:

- Váhy a bias se aktualizují na základě učícího koeficientu, chyby a hodnot vstupu.

Generování dat:

- Náhodně se generují 2D body, u kterých je předem určeno, zda leží nad či pod přímkou.

1.3 Implementace a vizualizace

Kód:

- Obsahuje funkce pro generování dat, výpočet lineární kombinace, aktualizaci vah a vyhodnocení přesnosti.

Vizualizace:

- Levý horní graf zobrazuje náhodně generované body a přímku.
- Pravý horní graf ukazuje rozhodovací hranici perceptronu.
- Spodní grafy znázorňují změny vah, biasu a přesnosti během tréninku.

1.4 Výsledky

Výsledky:

- Perceptron úspěšně rozdělil data podle zadané funkce a vývoj vah a biasu ukázal konvergenci modelu.

2. XOR Problem

2.1 Úvod a cíl

Tato úloha řeší klasický problém XOR, který nelze vyřešit jednoduchým perceptronem. Cílem je vytvořit plně propojenou neuronovou síť s jednou skrytou vrstvou, která dokáže XOR úspěšně klasifikovat.

2.2 Metodologie

Architektura:

- Vstupní vrstva: 2 neurony
- Skrytá vrstva: 2 neurony (s využitím sigmoidní aktivační funkce)
- Výstupní vrstva: 1 neuron

Učení:

- Používá se forward propagation k výpočtu výstupů.
- Chyba se počítá pomocí MSE (Mean Square Error).
- Backpropagation upravuje váhy a bias, kdy se chyba propaguje od výstupní vrstvy zpět do skryté vrstvy.

2.3 Implementace a vizualizace

Kód: Obsahuje implementaci Multi-Level Perceptronu s inicializací náhodných vah a biasů.

- **Trénink:** Síť prochází mnoha epochami, při kterých se váhy optimalizují a chyba se snižuje.
- **Výstup:** Finální váhy a biasy potvrzují, že síť správně řeší XOR problém.

Vizualizace:

- Graf evoluce celkové chyby v průběhu epoch.
- Grafy ukazující změny predikovaných výstupů, vah a biasů během tréninku.

2.4 Výsledky

Výsledky:

- Síť úspěšně řeší nelineární problém XOR, což potvrzuje správnost implementace backpropagation.

3. Hopfield Networks

3.1 Úvod a cíl

Cílem této úlohy je implementovat Hopfieldovu síť, která slouží k ukládání a následné obnově vzorů. Úloha zahrnuje implementaci dvou metod obnovy – synchronní a asynchronní.

3.2 Metodologie

Teoretické pozadí: Hopfieldova síť je rekurentní síť, kde každý neuron je propojen se všemi ostatními.
Trénink:

- Vytvoření paměťové matice pomocí vnějšího součinu (outer product) jednotlivých vzorů s následným nastavením diagonály na nulu.
  Obnova vzoru:

- **Synchronní obnova:** Všechny neurony se aktualizují najednou po dokončení celé iterace.

- **Asynchronní obnova:** Neurony se aktualizují postupně, což může ovlivnit dynamiku konvergence.

3.3 Implementace a vizualizace

Kód: Implementace využívá grafické rozhraní (Tk) pro zobrazení matice reprezentující Hopfieldovu síť a provádí obnovu vzorů.
Vizualizace:

- GIF ukazuje synchronní obnovu vzoru u mřížky (např. 5×5).

- Možnost interaktivních úprav vzorů a sledování procesu obnovy.

3.4 Výsledky

Výsledky:

- Síť byla úspěšně natrénována a dokáže obnovit uložené vzory. Lze vidět rozdíly mezi synchronní a asynchronní obnovou - první jmenovaná má tendence “halucinovat”.

4. Q-Learning

4.1 Úvod a cíl

Cílem této úlohy je implementovat Q-learning, který umožňuje agentovi (v našem případě myš) najít sýr v prostředí s překážkami (zdi, pasti). Agent se učí optimální politiku chování tak, aby maximalizoval odměny a minimalizoval penalizace.

4.2 Metodologie

Popis prostředí:

- Prostředí je reprezentováno jako mřížka (N×N) s rozmístěnými překážkami, pastmi, sýrem a agentem.

Akční možnosti:

- Agent se může pohybovat nahoru, dolů, doleva a doprava o jedno políčko.

Učení:

- Agent kombinuje strategii explorace a exploitace; s pravděpodobností epsilon vybírá náhodnou akci a jinak volí tu nejlepší.
- Q-matrix se aktualizuje podle rovnice, která kombinuje současnou odhadovanou hodnotu a získanou odměnu.

4.3 Implementace a vizualizace

Kód:

- Implementace zahrnuje inicializaci Q-matrix, funkce pro aktualizaci hodnot a výběr akcí. Využívá grafické rozhraní (Tk) pro zobrazení herního pole.

Trénink:

- Agent prochází mnoha kroky, během kterých se Q-matrix optimalizuje.

Výstup:

- Finální Q-matrix ukazuje hodnoty jednotlivých akcí, které pomáhají agentovi najít optimální cestu ke sýru.

Vizualizace:

- GIF ukazující tvorbu herního pole, trénink a úspěšné nalezení sýra.

- Grafické znázornění Q-matrix po tréninku.

4.4 Výsledky

Výsledky: Agent se postupně naučil najít nejefektivnější cestu ke sýru, vyhýbat se překážkám a snižovat počet kroků. Q-matrix demonstruje, že se agent adaptuje na dynamické prostředí a zlepšuje své rozhodování.

5. Pole-balancing problem

6. L-systems

5.1 Úvod a cíl

Cílem této úlohy je vytvořit interaktivní aplikaci pro generování a vizualizaci L-systému, který se používá pro vykreslování fraktálních struktur a modelování růstu rostlin. Aplikace umožňuje uživateli zadat počáteční axiom, produkční pravidlo, počet iterací a úhel otáčení. Na základě zadaných parametrů je vygenerován řetězec L-systému, který je následně interpretován do grafických příkazů pro vykreslení obrazce.

5.2 Metodologie

- **Generování řetězce:**
  Funkce generate_l_system rekurzivně aplikuje produkční pravidla na počáteční axiom. Každý symbol řetězce je nahrazen odpovídajícím pravidlem po dobu zadaného počtu iterací, čímž vzniká složitý řetězec příkazů.

- **Interpretace příkazů:**
  Funkce draw_l_system převádí vygenerovaný řetězec na geometrické operace.
  • Symbol "F" znamená posun vpřed se vykreslením čáry.
  • Symboly "+" a "-" představují otočení směru o zadaný úhel.
  • Symboly "[" a "]" zajišťují větvení, přičemž zásobník ukládá a obnovuje aktuální stav (pozici a úhel).

5.3 Implementace a vizualizace

- **Kód:**
  • Generování L-systému: Funkce generate_l_system(axiom, rules, iterations) zajišťuje postupnou expanzi řetězce pomocí předem definovaných pravidel (např. "F" -> "F+F").
  • Interpretace a vykreslení: Funkce draw_l_system(commands, angle_deg, step) interpretuje příkazový řetězec a na základě geometrických výpočtů vytváří sekvence bodů, které reprezentují jednotlivé větve fraktálu.

5.4 Výsledky

- Generování fraktálu: Aplikace úspěšně generuje komplexní fraktální struktury, jejichž vzhled se výrazně mění v závislosti na zadaném axiomu, produkčních pravidlech, počtu iterací a úhlu otáčení.

- Správnost interpretace: Použitím zásobníku pro větvení je dosaženo korektní interpretace příkazů, což se odráží ve správném vykreslení větvených obrazců.

- Interaktivita: Uživatelské rozhraní umožňuje snadné experimentování s parametry, což přispívá k lepšímu pochopení principů L-systémů a jejich využití při modelování přírodních struktur.

- Celkové vyhodnocení: Implementace demonstruje, jak jednoduché pravidla a rekurzivní algoritmy mohou vést k vytváření složitých a esteticky zajímavých geometrických obrazců, čímž potvrzuje funkčnost a flexibilitu přístupu.