УДК 004.8

ВИКОРИСТАННЯ МАШИННОГО НАВЧАННЯ В РОЗРОБЦІ ІГРОВИХ МЕХАНІК

Старобор Ігор Олегович

Науковий керівник: кандидат фізико-математичних наук,

Гуртовий Юрій Валерійович

Центральноукраїнський державний педагогічний університет імені Володимира Винниченка, м. Кропивницький, Україна

Анотація. *У статті розглядаються особливості використання такої підсистеми штучного інтелекту, як машинне навчання та її використання в ігрових додатках. Розглянуто Core ML, продемонстровано приклад його використання, який може полегшити бізнес логіку ігрового продукту. Проаналізовано плюси та мінуси фактори застосування даної системи в побудові ігрових механік. Описано поетапну роботу механіки. Виявлені переваги цієї платформи для штучного інтелекту в порівнянні з іншими схожими системами.*

Ключові слова: Машинне навчання, штучний інтелект, Core ML, Create ML, Ігри.

Using of machine learning in the development of game mechanics

Starobor Igor Olegovich

Supervisor: Candidate of Physics and Mathematics Sciences

Gurtovy Yuri Valerevich

The Volodymyr Vynnychenko Central Ukrainian State Pedagogical University, Kropyvnytskyi, Ukraine

Abstract. *The article considers the peculiarities of the use of such a subsystem of artificial intelligence as machine learning, it is using in game applications. Core ML is considered shown and an example of its application, which can facilitate the business logic of a gaming product. The pros and cons of using this system in the construction of game mechanics are considered, too. Described step-by-step work of mechanics. Revealed the advantages of this platform for artificial intelligence in comparison with other similar systems.*

Keywords: Machine Learning, Artificial Intelligence, Core ML, Create ML, Games.

Постановка проблеми. Однією з найважливіших областей серед дисциплін, що вивчають структуру і загальні властивості інформації є одна з підгалузей

штучного інтелекту – машинне навчання (Machine Learning). Стрімкий розвиток та поширення цифрових пристроїв дали поштовх до популяризації такої сфери, як машинне навчання, що дає можливість вирішувати такі типи завдань: побудова прогностичних рішень, розпізнавання цифрових даних в медіа файлах, їх класифікація, аналіз та багато іншого. Усе це можна використовувати у розширені та спрощені елементів геймдизайну. AI (Artificial intelligence - широка галузь комп'ютерних наук, які спрямовані на імітацію інтелекту людини машинами) та ML вже стали важливою частиною геймдеву, часто AI замінює процедурну генерацію чи використовується у оптимізації існуючих алгоритмів. Дедалі більше компаній стали впроваджувати в свої системи штучний інтелект, або машинне навчання для розвитку бізнесу. Завдяки цьому виник попит на розробку додатків з такими підсистемами.

Часткова, або повна автоматизація – це мета машинного навчання. Сфера застосувань машинного навчання постійно розширюється, глобальна комп’ютеризація стала причиною накопичення масивних обсягів даних, які складно обробляються звичайними алгоритмами. З одного боку використання даної технології є гарним рішенням для розв’язування такого роду проблем, але це має і недоліки, наприклад: довгий час навчання моделей, об’ємна кількість даних для розвитку систем. Результат готової моделі суттєво залежить від вхідних даних, на основі яких ведеться навчання. Дані можуть бути некоректними чи перекрученими - це може відбуватися як випадково, так і навмисно, що в подальшому дає хибний та неочікуваний результат роботи моделі машинного навчання.

Незважаючи на всі мінуси, дана тема є досить актуальною і вагомою для вивчення та дослідження моделей ML в комп’ютерних системах.

Аналіз останніх досліджень і публікацій. Ян Гудфеллоу, Іошуа Бенджі, Аарон Курвилль, одні з основних дослідників машинного навчання. Вони зробили великий вклад в цій сфері в своїй книзі “Глибоке навчання” (Deep Learning). Вона

описує методи глибокого навчання, що використовуються практиками в галузі. Ця книга пропонує вирішення інтуїтивних проблем у різних сферах. Ці рішення дозволяють комп'ютерам вчитися на досвіді та розуміти світ з точки зору ієрархії понять, причому кожна концепція визначається з точки зору її зв'язку з більш простими концепціями. Ієрархія понять дозволяє комп'ютеру вивчати складні поняття, будуючи їх з більш простих. З цієї причини автори називають такий підхід "глибоким навчанням". Що ж до використання машинного навчання в ігровій індустрії, оскільки галузь досить нова, велика частина інформації знаходиться в блогах, професійних ресурсах та документаціях.

Мета статті. Розглянути особливості та продемонструвати способи використання машинного навчання в геймдеві засобами Core ML на прикладі реалізації механік гри, сформованими за допомогою штучного інтелекту.

Методи дослідження. Для досягнення мети було використано такий емпіричний метод, як експеримент та теоретичні методи: аналіз та узагальнення інформаційних джерел і результатів експерименту.

Виклад основного матеріалу дослідження. Core ML - це основа для доменних фреймворків та функціональних можливостей. Core ML підтримує Vision для аналізу зображень, Natural Language для обробки тексту, Speech для перетворення аудіо в текст та Sound Analysis для ідентифікації звуків в аудіо. Сама Core ML будується на основі низькорівневих примітивів, таких як Accelerate та BNNS, що вбудована в екосистему Apple. [2]

Ще одна причина, через яку Create ML така популярна, полягає в її простоті використання. Вона набагато простіша, ніж інші популярні інструменти, такі як Tensorflow і Caffe. Ці інструменти вимагають великої кількості коду і не мають дружнього візуального інтерфейсу.

Створення ML використовує інфраструктуру машинного навчання, вбудовану в такі продукти Apple, як Photos та Siri.

Рис.1. Життєвий цикл навчання машинної моделі.

В ігровій індустрії даний метод може застосовуватися різними способами, наприклад: в створенні гри для пошуку об'єктів в просторі. При цьому методі навчена модель машинного навчання - Image Classifier класифікує вхідні дані з камери [4][5]. Потім їх можна аналізувати та обробляти у власних цілях.

Суть експерименту полягає в реалізації моделі штучного інтелекту, яка, спираючись на нові вхідні попередньо невідомі параметри, видасть визначаючу властивість, отримання якої допоможе знайти потрібний об'єкт, на основі аналізу результатів.

Для експерименту взято таку модель ML як регресор[3], який може передбачати числові значення, що виходять за рамки його навчальних даних. Що стосується розробки ігор, використання машинного навчання допомагає в спрощенні:

1. Базової ігрової логіки, наприклад зменшення обробки вхідних даних та їх валідація. Універсальне використання моделі надає можливість не застосовувати велику кількість шаблонного коду, в порівнянні з логіками без моделей машинного навчання.
2. Зменшенні кількості коду та часу реалізації. Його рев’ю проводити значно простіше, також знаходження недоліків чи помилок полегшується в рази. Від цього зменшується кількість створених об’єктів чи моделей даних, що сприяє швидкій розробці та спрощує орієнтацію в проекті. Тому навіть новим розробникам буде простіше підключитись до роботи над продуктом, який містить подібний функціонал. Така модель ML

може бути локальною, тому вона не потребує в створенні мережевої логіки, шару інтерфейсів для обробки даних по принципу загальноприйнятої клієнт-серверної архітектури.

Проте, є і негативні ефекти від використання цієї технології:

1. Модель є закінченою і не підлягає обробці в майбутньому. Core ML будується завчасно для подальшого використання в програмі. Такий підхід має свої наслідки: розвиток поточної моделі неможливий, тому єдиним варіантом редагування моделі лишається оновлення з альтернативних джерел, наприклад сервер, який формує оновлення на основі відправлених даних з пристрою. До того ж її створення на локальному пристрої є недоцільним та ресурсозатратним. Якщо вона використовується на різних девайсах, цей підхід доречний.
2. Точність результатів, але це залежить від очікуваних даних. В машинному навчанні точність не найважливіша складова, вона спрощує роботу, а не збільшує досконаліть результатів, тому її не використовують в розробці програмного забезпечення, яке потребує надзвичайно чітких даних.

Щоб перевірити ефективність цієї підсистеми штучного інтелекту в розробці ігор, був проведений експеримент.

Генерація даних стала першим кроком для створення експериментальної моделі ML. Так, як знайти відповідні відкриті джерела, де надають дані згідно вимог поставленої задачі неможливо, було розроблено генератор псевдоданих, який створив правильно сформований csv-файл, що має параметри гравців, де одним з параметрів є його рейтинг. Як вище зазначено, в експерименті використовується табличний регресор він приймає на вхід тільки дані числового типу, які розділені між собой відповідними символами для представлення в табличному вигляді. В кінечному результаті формується такий набір параметрів: vSp - швидкість транспортного засобу, vAg - вік транспортного засобу, vWe - вага транспортного засобу, vSt - стан транспортного засобу, vEx - пробіг

транспортного засобу, rAg - вік водія, rWe - вага водія, rEx - досвід водія, rFi - показник ресурсів водія, gDi - потенціал гравця на певній дистанції, gWe - потенціал гравця за певних погодних умов, gTr - потенціал гравця на певному типі дороги , gTi - потенціал гравця в певну пору доби, tDi - довжина дистанції, tWe - погодні умови дистанції, tSt - стан дистанції, tTi - пора доби на дистанції. На основі них штучний інтелект буде намагатися дати остаточний результат, що і є останнім та ключовим параметром - rating.

Наступним кроком, після перевірки коректності створених даних, є їх передача в Create ML, що на їх основі створить модель машинного навчання, яку можна буде використовувати для такої цілі, як отримання рейтингу гравця з випадково згенерованих даних під час гри. Якщо набір наданих даних має валідну структуру та правильний набір параметрів які містять регламентований тип, то система прийме їх та надасть запит на поля що будуть використані в процесі навчання регресора. Обравши всі параметри та цільову вихідну властивість ми можемо розпочати процес навчання. Залежно від об'єму даних та ресурсів машини на якій відбувається процес, час витрачений на це може варіювати в значних діапазонах. В кінцевому вигляді система видасть структуру моделі готову до використання. Вона буде зберігатись локально на пристрої в скомпільованому додатку, таким чином швидкості доступу та отримання результатів будуть моментальними, а незначна вага моделі не буде суттєво впливати на загальну вагу програми.

Фінальний крок експерименту - тестування моделі, в якому на основі згенерованих даних підбираються відповідні асоціації для користувача. Для випробування моделі, описаної в другому кроці, ми генеруємо значення для параметрів, названих вище. Потім реалізується етап надання цих параметрів регресору. Зробимо декілька очевидно зрозумілих ситуацій, де ми очікуємо середній, високий та низький рейтинг. Далі створюються гравці з найбільшими, середніми та низькими параметрами. Штучний інтелект не володіє інформацією

про конкретно ці параметри, але навчений на тисячах подібних даних і повинен надавати результат приблизний до очікуваного, щоб асоціації відповідали дійсності та користувач міг вибрати максимально достовірний варіант з найвищим рейтингом Рис.2.

```
func selected(index: Int) {
    let model = HorseRacingCML()
    print(index)
    let selectedPlayer = players.data.horses[index]
    for (horseIndex, horse) in players.data.horses.enumerated() {
        guard let horsesOutput = try? model.prediction(hSp: horse.hSp, hAg: horse.hAg, hWe: horse.hWe,
            hSt: Double(horse.hSt?.rawValue ?? 0), hEx: horse.hEx, rAg: horse.rAg, rWe: horse.rWe, rEx:
            horse.rEx, rSt: Double(horse.rSt!.rawValue), rFi: horse.rFi, gDi: horse.gDi, gWe: horse.gWe,
            gTr: horse.gTr, gTi: horse.gTi, tDi: players.data.map.tDi, tWe: players.data.map.tWe, tTr:
            players.data.map.tTr, tTi: players.data.map.tTi) else {
            fatalError("Unexpected runtime error.")
        }
        players.data.horses[horseIndex].viktories = Int(horsesOutput.victories)
        print("player ID - \(players.data.horses[horseIndex].id)",
              "player rating - \(players.data.horses[horseIndex].viktories!)")
    }

    let winer = players.data.horses.max(by: {$0.viktories ?? 0 < $1.viktories ?? 0})

    print("\nSelected player ID - \(selectedPlayer.id)",
          "\nWiner player ID - \(winer!.id)")
    print(winer?.id == selectedPlayer.id ? "You win" : "You lose")
}
```

Рис.2. Функція отримання рейтингу та визначення переможця за обраним гравцем.

Це і буде основною ціллю гри. Отримавши результати, проведемо ще один дослід, в якому згенеруємо більше варіантів гравців, щоб конкретизувати і впевнитися в тому, що ми отримали очікуваний висновок Рис.3.

```
player ID - E6920557-CF25-429A-9911-71EF29928DA3 player rating - 49
player ID - C0F94B44-E5AA-4AEC-AFD5-CDA11D3C172C player rating - 44
player ID - ECAB2FB9-D5C5-4D7B-AF73-1C0891338E3D player rating - 18
player ID - C1AEAF13-BFE0-4E3A-A7AD-4150D090836C player rating - 53
player ID - 5688E777-B28D-451B-A11C-883EF6A29163 player rating - -11

Selected player ID - C1AEAF13-BFE0-4E3A-A7AD-4150D090836C
Winer player ID - C1AEAF13-BFE0-4E3A-A7AD-4150D090836C
You win
```

Рис.3. Приклад отриманих даних з вибором переможця.

Отже після проведення експерименту можна зробити висновки, що створена модель є компактною та займає всього 1 кілобайт інформації. Покращує читабельність коду та спрощує логіку написаної програми тим, що для отримання результатів достатньо написати всього лиш одну невелику функцію. Похибка результатів не є важливим аспектом в досягненні нашої мети, тому очікувані дані повністю задовольняють потреби ігрової логіки. При отриманні результатів з декількох експериментів ми визначили, що вони близькі до очікуваних.

Список джерел.

1. Ян Гудфеллоу, Иошуа Бенджио, Аарон Курвилль. Глубокое обучение. — М.: ДМК Пресс, 2017. — 652 с.:ил. ISBN 978-5-97060-618-6.
2. Apple [Електронний ресурс] : (документація) / Core ML – 2020. Режим доступу: https://developer.apple.com/documentation/coreml – Назва з екрану
3. Apple [Електронний ресурс] : (документація) / Creating a Model from Tabular Data – 2020. Режим доступу: https://developer.apple.com/documentation/createml/creating_a_model_from_tabular_data – Назва з екрану
4. Apple [Електронний ресурс] : (документація) / Classifying Images with Vision and Core ML – 2020. Режим доступу: https://developer.apple.com/documentation/vision/classifying_images_with_vision_and_core_ml – Назва з екрану
5. Heartbeat [Електронний ресурс] / Anupam Chugh // Build a SwiftUI + Core ML Emoji Hunt Game for iOS – 2019. Режим доступу: https://heartbeat.fritz.ai/build-a-swiftui-core-ml-emoji-hunt-game-for-ios-eb4465ec4153 – Назва з екрану
6. Apple [Електронний ресурс] : (документація) / Apple Developer Documentation. Режим доступу: https://developer.apple.com/documentation/ – Назва з екрану
7. ZDNet [Електронний ресурс] / Nick Heath // What is machine learning? Everything you need to know. Режим доступу: https://www.zdnet.com/article/what-is-machine-learning-everything-you-need-to-know/ – Назва з екрану