Мальченко П. О.,
асистент кафедри економічної кібернетики,
комп'ютерних наук та інформаційних технологій
Миколаївський національний аграрний університет, м. Миколаїв

СТВОРЕННЯ РЕКУРЕНТНИХ НЕЙРОННИХ МЕРЕЖ ЯК ЗАСІБ ПОКРАЩЕННЯ ЗНАНЬ СТУДЕНТІВ У СФЕРІ РОБОТИ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

Рекурентна природа рекурентних штучних нейронних мереж (RNN) дозволяє програмам ШІ фіксувати часові залежності та генерувати когерентні послідовності даних. У RNN ваше передбачення залежить не тільки від особливостей поточного моменту часу, але й від моментів часу, що передували поточному. Як наслідок, ці ШНМ найбільш корисні для моделювання закономірностей у послідовностях даних, де наступний елемент залежить від попередніх елементів у послідовності.

Рекурентні нейронні мережі використовуються для різноманітних задач: генерація послідовностей, прогнозування часових рядів, генерація музики, створення підписів до зображень, генерація діалогів, генерація рукописного тексту, опис відео, генерація коду [1].

Особливо цікавим є можливості рекурентних мереж до генерації діалогу. RNN, особливо у вигляді моделей "послідовність-послідовність" (seq2seq), широко використовуються в задачах генерації діалогів. Ці моделі складаються з архітектури кодер-декодер. Кодер RNN обробляє вхідну послідовність (наприклад, запит або повідомлення користувача) і кодує її у вектор контексту фіксованої довжини, фіксуючи значення і контекст вхідних даних. Потім декодер RNN генерує відповідь на основі цього вектора контексту. Для навчання чат-бота або віртуального асистента на основі RNN зазвичай використовується великий набір розмовних даних. Цей набір даних складається з пар вхідних повідомлень і відповідних відповідей. RNN навчається передбачати відповідну реакцію на вхідне повідомлення, навчаючись генерувати відповіді, подібні до людських, на основі шаблонів у навчальних даних. Стандартні RNN можуть мати проблеми з фіксацією довготривалих залежностей у послідовностях, що має вирішальне значення для побудови зв'язного діалогу. LSTM та GRU-варіанти RNN вирішують цю проблему шляхом включення механізмів, які вибірково зберігають або забувають інформацію в довгих послідовностях. Це дозволяє їм краще вловлювати контекст і генерувати більш змістовні відповіді в розмовах. На додаток до LSTM або GRU-комірок, в моделях генерації діалогів часто використовуються механізми уваги. Увага дозволяє моделі фокусуватися на різних частинах вхідної послідовності під час генерації кожного токена вихідної послідовності. Це допомагає вловлювати релевантний контекст з вхідної інформації, особливо в довгих розмовах або при відповіді на складні запити.

Зважаючи на такі потужні можливості RNN, важливо поліпшити знання людей про можливості їх створення та налаштування для зручного та ефективного користування. В рамках цієї роботи розглянуто створення моделі

для прогнозування часових рядів з погодними умовами, що можна вважати відправною точкою при вивченні способів проєктування RNN.

Для навчання моделі було використано набір даних з Kaggle з погодними умовами [2]. Цей набір даних містить інформацію про погоду в Сіетлі за кожен день з 1 січня 1948 року по 12 грудня 2017 року. Для кожного дня ми маємо кількість опадів, максимальну та мінімальну температуру, а також те, чи були опади взагалі в цей день (ми припускаємо, що зазвичай йшов дощ, і тому колонка називається "дощ").

Було встановлено три параметри:

• Максимальна довжина послідовності, яку ми будемо розглядати, щоб спробувати передбачити наступний елемент

• Розмір партії, тобто кількість різних послідовностей, які потрібно переглянути за один раз під час навчання (таким чином, загальна кількість точок даних, які ви будете вводити в модель за один раз, дорівнює максимальній довжині партії). Зазвичай бажано, щоб це був степінь двійки (наприклад, 16, 32, 64 і т.д.), щоб полегшити собі життя в майбутньому, якщо ви захочете тренувати модель на графічному процесорі.

• Загальна кількість епох для навчання, тобто кількість разів, коли модель буде піддаватися впливу всього навчального набору (таким чином, кількість раундів навчання, які наша модель буде тренувати = (загальна кількість навчальних прикладів/розмір пакету) * загальна кількість епох навчання).

Можна сказати, що для виконання цього завдання використання глибинного навчання є надлишковим, але це гарний приклад для розуміння самого принципу роботи рекурентних нейронних мереж.

Після перевірки навчальних даних на чистоту, наступне завдання – взяти цей вектор, а потім розбити його на відрізки довжиною max_length + 1 (оскільки ми хочемо знати, який наступний елемент у послідовності, щоб побачити, наскільки добре ми його передбачили: ми будемо відрізати його перед тим, як передавати навчальні дані нашій моделі). Ми могли б просто почати з початку нашого вектора і розбити його на шматки довжиною max_length + 1, що не перетинаються, але цього насправді недостатньо для навчання моделі глибокого навчання. Щоб розтягнути наші дані, ми можемо використати так звану субдискретизацію з рухомим блоком, де ми розрізаємо наш вектор на частини, що перекриваються. Хоча це схоже на інші методи підвибірки, дуже важливо, щоб ми не змінювали порядок елементів у наших блоках! Оскільки ми будемо навчати рекурентну нейронну мережу, яка моделює взаємозв’язок елементів у послідовності, нам важливий порядок наших спостережень, і ми хочемо переконатися, що він буде збережений.

Після цього було внесено дані в модель бібліотеки Keras. По-перше, було розділено набір даних на вхідні (шість попередніх днів) і вихідні (один день, який потрібно спрогнозувати). Далі було вказано, як має виглядати модель. На цьому кроці Keras дійсно блищить. Він був розроблений для того, щоб можна було вказати, як має виглядати модель на дуже високому рівні, описуючи кожен шар моделі в загальних рисах.

Перед початком навчання потрібно повідомити Keras, скільки шарів повинна мати модель і яким має бути кожен шар. Архітектура моделі виглядатиме наступним чином:

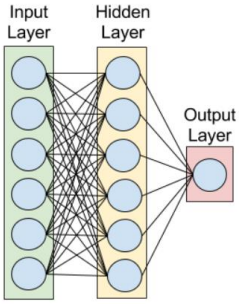

Рис. 1. Архітектура моделі [2]

Залишився останній крок у налаштуванні моделі, перш ніж можна буде її навчати. Потрібно скомпілювати її та вказати, якими мають бути втрати та оптимізатор, а також яку метрику відстежувати.

Після навчання можна побачити, як значення змінювалися під час навчання моделі, будуючи графік на основі історії, яку зберегли раніше. Тут видно, що точність зросла, а втрати зменшилися як для навчальних, так і для валідаційних даних з плином часу:

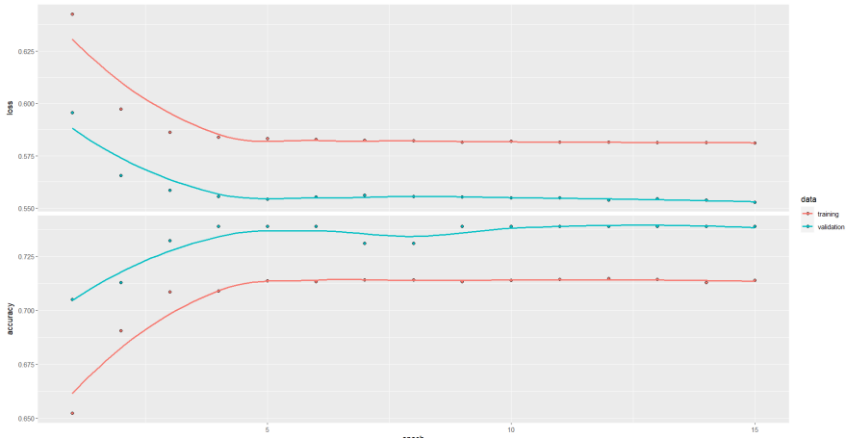

Рис. 2. Зміна продуктивності під час навчання

Доцільно було б припинити тренування на кілка епох раніше, оскільки і точність, і втрати вирівнялися приблизно на 5 епосі.

Список використаних джерел

1. K. Niwayama, K. Muto and Y. Kobayashi, "Convolutional Recurrent Neural Network-Based Boat Detection Method for Wind Noise Condition," 2022 IEEE 11th Global Conference on Consumer Electronics (GCCE), Osaka, Japan, 2022, pp. 3-4, doi: 10.1109/GCCE56475.2022.10014118.

2. Beginner’s Intro to RNN’s in R. https://www.kaggle.com/code/rtatman/beginner-s-intro-to-rnn-s-in-r/notebook