ISSN 2078–5097. Вісн. Львів. ун-ту. Сер. прикл. матем. та інф. 2018. Вип. 26. С. 118–123
Visnyk of the Lviv University. Series Appl. Math. and Informatics. Issue 26. P. 118–123

УДК 004.4

ПІДХІД ДО ДИНАМІЧНОГО ДОНАВЧАННЯ ДЕРЕВ РІШЕНЬ

М. Гривняк, Ю. Щербина

Львівський національний університет імені Івана Франка,
вул. Університетська, 1, Львів, 79000,
e-mail: martahryvniak@gmail.com; yshcherbyna@yahoo.com

Зроблено спробу знайти узагальнений підхід до динамічного донавчання дерев рішень. З використанням алгоритмів Accord .NET Framework розроблено веб-додаток, здатний будувати, зберігати, добудовувати, тестувати та виводити дерева рішень зручним способом. За допомогою веб-додатка запропонований підхід протестовано та зроблено висновки про його доцільність і сферу застосування.

Ключові слова: машинне навчання, граф, дерево рішень, класифікація, похибка.

1. ВСТУП

Машинне навчання – це напрям штучного інтелекту. Це не просто тема для наукових досліджень, а потужний інструмент у багатьох додатках і сервісах. Без цього інструмента багато відомих зараз сервісів не змогли б впоратися з конкуренцією на ринку.

Найбільш несподіваним наслідком переходу від програм на локальному комп'ютері до веб-сервісів стало те, що ми можемо автоматично збирати величезні масиви даних користувачів. Завдяки можливостям впроваджувати поліпшення та негайно впливати на користувачів інтернет-гіганти, Фейсбук чи Гугл, активно працюють над підходами до опрацювання великих масивів даних. Іншим помітним явищем стало стрімке зростання та здешевлення обчислювальних потужностей у світі. Саме ці особливості є передумовами, які штовхають машинне навчання (МШ) вперед.

Компанія Microsoft підтримує загальні тенденції демократизації машинного навчання та дає можливості для розробки з використанням моделей, які здатні до навчання, для адептів .NET Framework і не тільки. Тепер майже кожна мова програмування має набір з бібліотек і класів, які дають базовий функціонал для роботи з основними алгоритмами, і .NET Framework тут не виняток. Зокрема, цікавим є Accord .NET Framework як розвинений продукт з відкритим вихідним кодом.

Більшість алгоритмів для побудови дерева рішень не призначені для донавчання. Після побудови моделі вони не можуть поліпшуватися на підставі додаткових даних, а лише перебудовуватися. Можливість добудовувати дерева рішень для деяких задач могла би дати значні переваги.

Запропонуємо можливий підхід до динамічного добудовування дерев рішень. Мета підходу – оптимізувати споживання пам'яті на жорсткому диску в умовах великої кількості даних, або обмеженої пам'яті. Іншою перевагою є можливість збільшення продуктивності, підвантажуючи нову порцію даних. Позаяк дерева рішень слугують для вирішення реальних задач, їхнє динамічне добудовування розкрило б перед нами можливість динамічної (ітеративної) оптимізації процесів і рішень.

2. ФОРМУЛЮВАННЯ ЗАДАЧІ

Задача динамічного дерева рішень полягає в тому, щоб у ситуації, коли отримуємо дані не одразу, а порціями, ми могли побудувати перше дерево, отримавши хоча б одну досить велику порцію даних. Коли надходить наступна порція даних, ми вже не можемо звернутися до попередньо отриманих наборів. Натомість, як основу для побудови наступного дерева рішень використовуємо дерево, побудоване на їхній основі.

Так задається вимога повною мірою скористатися однією з трьох основних функцій дерева рішень – функцією “опису даних”. Дерево слугує своєрідним “архівом” для даних, воно допомагає зберігати інформацію про дані у компактному вигляді. На відміну від алгоритмів стискання даних, використання дерева рішень призводить до великої втрати інформації, але послуговуючись ним мають на меті зберегти максимум інформації про вплив атрибутів на значення вихідної змінної.

Існує багато переваг використання донавчання.

- Результати аналізу доступні одразу, а отже, можуть використовуватися як підстава для висновків і дій – давати змогу динамічній оптимізації виробництва/лікування і т.п.
- Проміжні дані не зберігаються на жорсткому диску, що допомагає зменшити споживання пам’яті в умовах величезних кількостей даних або обмежених ресурсів.
- Донавчання дерева на основі нової порції даних може бути ефективнішим/швидшим, ніж об’єднання з усіма попередніми даними та навчання на усіх даних, за умови розробки ефективного опрацювання даних.

3. ПІДХІД ДО РОЗВ’ЯЗАННЯ ЗАДАЧІ

Щоб розробити підхід, який буде сумісний з такими алгоритмами побудови дерева рішень як C4.5 та ID3, потрібно щоразу подавати на вхід таке подання даних, яке буде сумісне з кожним з цих алгоритмів. З першим набором даних ми можемо працювати у звичному режимі. Надалі, щоб отримати подання дерева у вигляді сумісному з новим набором даних, використовуємо розгортання дерева, що полягає у поданні побудованого дерева вибіркою даних. Щоб під час розкладання дерева, якнайкраще наблизитися до вихідної вибірки даних, було організовано зберігання додаткової інформації, що не передбачалося класичними алгоритмами.

Одна з метрик, які було вирішено зберігати додатково – кількість даних. Кожен шлях від кореня до листка можна зіставити з певною кількістю кортежів, що його породили. Інформація про кількість даних на кожному з листків має зберігатися додатково з листком, щоб розуміти “силу” того чи іншого правила. Отож, кожному листку ставимо у відповідність число – кількість кортежів, що його породили.

Крім кількості даних, треба мати уявлення про значення атрибутів, які породили листок. Тут натрапляємо на проблему, що самого лише дерева недостатньо, щоб отримати уявлення про значення усіх атрибутів. Адже не обов’язково усі атрибути беруть участь у кожному шляху від кореня до листка. Крім того, навіть якщо кожен з атрибутів взяв участь, то ми отримаємо обмеження значення лише з одного боку (> або <). Тому додатково для кожного листка зберігатимемо середні значення атрибутів з тієї частини набору даних, яка його породила.

У дереві рішень у випадку наявності суперечливих значень вихідної змінної на листку, перевага надається тому виходу, який трапляється найбільшу кількість разів. Так закладається похибка на листку. Алгоритми побудови дерева рішень можуть зберігати значення цієї похибки. Так ми знатимемо, який відсоток відповідної порції даних на листку припадав на значення вихідної змінної. Але тоді ми втратимо інформацію про ті значення вихідної змінної, які були в меншості. Для того, щоб уникнути цього, будемо зберігати відповідно до кожного листка не лише значення вихідної змінної, яке трапляється найбільшу кількість разів, а й усіх інших значень разом з кількістю разів, коли значення з'являлося. Якщо середні значення атрибутів будуть однаковими для різних можливих вихідних даних, то ці дані не зможуть вплинути на формування видозмінених шляхів у подальшому, а лише на вибір вихідного значення. Тож доцільно буде зберігати середні значення атрибутів не відповідно до листка, а відповідно до кожного можливого виходу на кожному з листків.

Отож, до кожного листка додається набір об'єктів, що містить такі обов'язкові поля:

- посилання на листок;
- індекс значення вихідної змінної;
- посилання на вхідний атрибут;
- середнє значення вхідного атрибуту;
- кількість кортежів у тренувальному наборі даних, які відповідали б вищенаведеним характеристикам.

Якщо N – це кількість листків дерева; M – кількість атрибутів; а K – кількість можливих значення вихідної змінної, тоді кількість таких об'єктів, які потрібно додатково зберігати, становитиме:

$P = NMk,\ де\ 1 \le k \le K;$

$NM \le P \le NMK.$

N, M, K – цілі числа.

Тепер, коли на вхід приходить новий набір даних, можемо розкласти попередньо побудоване дерево у набір даних, де кожен кортеж відповідає вищезгаданому об'єкту так, що значення атрибутів беремо з їхніх середніх значень. Там, де кількість даних більше нуля, такі кортежі будуть дублюватися відповідну кількість разів. На практиці не треба генерувати в оперативній пам'яті набір даних з великою кількістю дублікатів. Можна модифікувати алгоритм побудови дерева рішень, щоб він разом з простими даними приймав на вхід ще й агреговані дані.

4. ТЕСТУВАННЯ НА НАБОРАХ ДАНИХ

Для тестування вищенаведеного підходу було розроблено веб-додаток з базою даних з використанням таких технологій: HTML/CSS/JQuery, ASP.NET Web API, Accord .NET Framework, Microsoft SQL Server.

Усі набори даних, на яких проводили тестування, взяли з наборів даних розміщених у відкритому доступі на сайті Каліфорнійського Університету у місті Ірвін [2]. Далі наведемо результати тестування на двох наборах даних: "Іриси Фішера" (англ. *Iris flower data set*) та "Діабетична ретинопатія" (англ. *Diabetic Retinopathy Debrecen Data Set*).

Обсяг класичного набору даних “Іриси Фішера” становить 150 кортежів. Кожен кортеж характеризується чотирма неперервними вхідними атрибутами та однією дискретною вихідною змінною з трьома можливими значеннями.

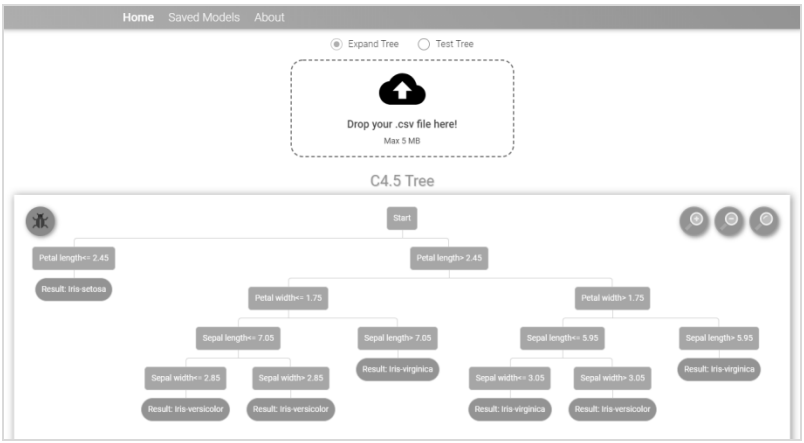

Рис. 1. Інтерфейс веб-додатку

У машинному навчанні прийнято тренувати модель на одному наборі даних, а тестувати на іншому. Щоб продемонструвати ефективність підходу до динамічного доповнення дерев рішень, запропонованого у цій праці, є сенс на кожній ітерації (додавання нової порції даних) тестувати результат на усіх попередньо переданих даних, аби проаналізувати скільки інформації збереглося, порівняно з деревом, яке було б побудоване на підставі всієї вибірки одразу без динамічного добудовування. З огляду на малий обсяг вибірки “Іриси Фішера” тестовий набір тут не виокремлювався.

Набір даних було розділено на дві порції позначені (1) і (2). Запропонований підхід призводить до втрати інформації і найбільше тієї, яку отримали з раніше переданих наборів даних. Найдетальніше інформація зберігається з того набору даних, які було застосовано найпізніше. Тому доцільно буде тестувати, враховуючи додавання даних у різному порядку.

Таблиця 1

Результати тестування на підставі набору даних “Іриси Фішера”

Кількість даних і порядок тренування дерева	Похибка при тестуванні на наборах даних		
	75 тренувальних записів (1)	75 тренувальних записів (2)	150 тренувальних записів
75 (1) + 75 (2)	0,067	0,013	0,040
75 (2) + 75 (1)	0,040	0,027	0,033
150	0,027	0,027	0,027

Бачимо, що при донавчанні з використанням порції даних з 75 записів, логіка добудовування дерева рішень працює доволі ефективно. Спостерігаємо збільшення

похибки на 0,006-0,013, що є незначним. У подальшому розбитті на три, чотири і більше порцій даних результуюча помилка перевищує 50%, що свідчить про неефективність цього підходу для заданого набору даних при менших обсягах разової порції даних, тому є сенс перейти до роботи з більшим набором даних.

Обсяг вибірки "Діабетична ретинопатія" становить 1151 кортеж, кожен з яких характеризується 20 вхідними атрибутами (фактично 19, бо перший є константою) та однією вихідною дискретною змінною зі значенням 0 або 1. Набір було розділено на 1000 кортежів тренувальних даних, які поділені на дві порції по 500, та 151 кортеж даних для тестування.

Таблиця 2

Результати тестування на підставі набору даних "Діабетична ретинопатія"

Кількість даних і порядок тренування дерева	Похибка при тестуванні на наборах даних			
	500 тренувальних записів (1)	500 тренувальних записів (2)	1000 тренувальних записів	151 тестових записів
500 (1) + 500 (2)	0,380	0,058	0,219	0,610
500 (2) + 500 (1)	0,074	0,412	0,243	0,623
1000	0,212	0,190	0,201	0,596

З наведеного прикладу видно, що цей підхід дає найкращу точність для останнього набору даних, який використовували для тренування. Для ефективності алгоритму на кожному етапі має надходити інформація, що містить всілякі можливі виходи. Недотримання цієї вимоги особливо відбивається на першій порції даних. Якщо у першій порції даних надійде інформація лише з одним вихідним атрибутом, то жодної інформації не буде збережено, крім середніх значень. Також обсяг порції даних має бути досить великим. Доповнення працює найефективніше, коли набори приблизно однорідні і кожна наступна ітерація лише уточнює попереднє дерево, але не добудовує його суттєво.

5. ВИСНОВКИ

На підставі тестування на двох наборах даних можна зробити висновок, що наведений підхід може бути ефективним. Проте він застосовний до вузького класу задач з накладенням умов на обсяг і зміст даних у кожній порції, яка передається алгоритму. Можна сказати, що спершу ми маємо побудувати досить інформативну модель, яка надалі буде лише уточнюватися. Якщо порції даних різноманітні між собою, але не всередині себе, то це призводить до втрат інформації з використанням запропонованого підходу.

Для ефективності алгоритму потрібні такі умови:

1) у порції даних мають надходити кортежі з усіма можливими виходами;
2) обсяг порції даних має бути досить великим (залежить від набору).

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. *Нікольський Ю.В.* Системи штучного інтелекту: навчальний посібник / Ю.В. Нікольський, В.В. Пасічник, Ю.М. Щербина. – Львів: "Магнолія-2006", 2013. – 279 с.
2. UC Irvine Machine Learning Repository. – 2018. – Режим доступу: http://archive.ics.uci.edu/ml/index.php
3. *Esmeir S.* Anytime Learning of Decision Trees / S. Esmeir, S. Markovitch // J. Mach. Learn. Res. – May. – 2007. – Vol. 8. – P. 891-933.

Стаття: надійшла до редколегії 12.09.2018
доопрацьовано 24.10.2018
прийнята до друку 31.10.2018

AN APPROACH TO DYNAMIC ADDITIONAL TRAINING OF DECISION TREES

M. Hryvniak, Yu. Shcherbyna

Ivan Franko National University of Lviv,
Universytetska Str., 1, Lviv, 79000,
e-mail: martahryvniak@gmail.com; yshcherbyna@yahoo.com

Most algorithms for constructing a decision tree are not intended to be additionally taught. After constructing the model, they cannot be improved on the basis of additional data, but only rebuild. The ability to improve the decision tree using additional data without rebuilding could give significant benefits in some cases. Such benefits as optimization memory consumption on the hard disk or increase in performance when loading a new portion of data. And since the decision trees are used to solve real problems, their dynamic improving would open up the possibility of dynamic (iterative) optimization of processes and solutions.

There was made an attempt to find generalized approach to dynamic additional training of decision tree ML models. There was web application developed to build, save, perform additional training, test and display decision trees in convenient way using algorithms from Accord .NET Framework. Using web application mentioned approach was tested and conclusions has been made about its expediency and area of usage.

As a result of testing on two sets of data, it can be concluded that this approach can be effective. However, it is applicable to a narrow class of problems with the imposition of conditions on the amount and content of data in each portion passed to the algorithm. We can say that at first, we have to build a quite informative model, which in the future will only be clarified. First portions should represent the whole set and contain the whole variety of outputs, otherwise the proposed approach leads to dramatically loss of information.

The following conditions are needed for the effectiveness of the algorithm:

1. In a portion of data, tuples should contain all possible outputs.
2. Size of the portion of data should be large enough (depending on the set).

Key words: machine learning, graph, decision tree, classification, error.