УДК 004.93,004.8
DOI: 10.18523/2617-3808.2019.2.22-31

Глибовець М. М., Жиркова А. П.

ВИКОРИСТАННЯ МАШИННОГО НАВЧАННЯ У ЗАДАЧАХ КЛАСИФІКАЦІЇ ЗВУКІВ

У роботі розглянуто особливості використання методів машинного навчання (МН) для класифікації звукової інформації на прикладі розв'язку задачі класифікації міських звуків (МЗ). Дослідження з аналізу міських акустичних середовищ є досить обмеженими. Більше того, у цих дослідженнях основна увага фокусується на класифікації місць, які характеризують певні звуки, наприклад, парку, вулиці, на відміну від ідентифікації джерел звуку в них, таких як автомобільний сигнал, постріл тощо. Тому класифікація МЗ – досить актуальна проблема, що потребує вирішення. Метою роботи є висвітлення побудови оптимальних моделей МН для задачі коректної класифікації МЗ.

Ключові слова: класифікація, машинне навчання, метод k-найближчих сусідів, метод опорних векторів, випадковий ліс, нейронні мережі, контроль за k-блоками, метод відкладених даних, сітковий пошук.

Вступ

Основною ідеєю машинного навчання (МН) є знаходження певних закономірностей у відомих наборах даних і навчанні машини передбачати результат для нових даних. Існує декілька видів задач МН, серед яких традиційно виділяють регресію, класифікацію, кластеризацію [9, p. 47, 307]. Регресія від класифікації відрізняється необхідністю знаходження певного числа, що відповідає об'єкту дослідження, в той час як задача класифікації ставить у відповідність об'єкту клас, до якого він належить. У випадку кластеризації задача полягає у групуванні різних об'єктів, надаючи кожному певний клас.

Існує три базові підходи до МН. Першим є навчання з учителем, що означає наявність цільової змінної у тренувальних даних (використовують для навчання конкретної моделі). У результаті алгоритм може зіставляти ознаки об'єкта з цільовою змінною і на основі цього передбачати її значення для інших об'єктів. Другий підхід – навчання без учителя – відрізняється від першого відсутністю цільової змінної і необхідністю її знаходження, тобто, потрібно передбачити. Третій – навчання з підкріпленням, коли результат будується у відповідь на «подразнення» зовнішнього середовища.

У цій роботі розглянуто задачу класифікації міських звуків за допомогою різних алгоритмів МН.

МН допомагає формалізувати знання експертів для вирішення різних задач із багатьох прикладних галузей застосувань, надаючи можливість працювати з різноманітними типами даних, у тому числі з текстом, зображеннями та звуками. Автоматична класифікація звуків навколишнього середовища – зростаюча сфера досліджень. Зокрема, звуковий аналіз міських середовищ є об'єктом підвищеного інтересу, частково завдяки мультимедійним сенсорним мережам, частково завдяки великій кількості мультимедійного контенту, що зображує міські сцени.

Однак, незважаючи на велику кількість досліджень у відповідних галузях, наприклад, у мові та музиці, дослідження з аналізу міських акустичних середовищ є досить обмеженими. Більше того, у цих дослідженнях основна увага фокусується на класифікації місць, які характеризують певні звуки, наприклад, парку, вулиці, на відміну від ідентифікації джерел звуку в них, таких як автомобільний сигнал, постріл тощо. Тому класифікація міських звуків – досить актуальна проблема, що потребує вирішення.

Розробка оптимальних моделей МН для задачі коректної класифікації міських звуків потребує вирішення таких завдань: представлення звуку як сукупності числових даних, тобто його ознак, для можливості роботи алгоритмів МН із ним; обробка даних для оптимізації ефективності роботи моделей МН; застосування методів оптимізації моделей МН задля знаходження оптимальних параметрів кожної з моделей; проведення навчання моделей на тренувальних даних; застосування моделей для передбачення результату та перевірки точності їх роботи на тестових даних; розробка моделі нейронної мережі для вирішення задачі класифікації міських звуків.

1. ОСОБЛИВОСТІ ЗАСТОСУВАННЯ МН ДЛЯ РОБОТИ ЗІ ЗВУКОМ

1.1. Аналіз існуючих алгоритмів МН

Під час розв'язання будь-якої задачі за допомогою МН передусім треба обрати алгоритм, який буде її вирішувати. До того ж, варто зауважити, що кожен із них орієнтований на розв'язання певного типу задач.

Найбільш простий і популярний – лінійна регресія [9, p. 123]. Суть цього алгоритму полягає в зображенні кожної точки у вигляді лінійного рівняння, де ознаки (features) цієї точки – незалежні змінні, ваги (weights) – коефіцієнти, а шукане значення (target) – залежна змінна. На етапі навчання підбирають параметри моделі (в цьому алгоритмі – ваги) таким чином, щоб лінія, отримана в результаті їхнього підбору, проходила якомога ближче до всіх точок. При безпосередньому прогнозуванні ознаки та обрані параметри підставляють у рівняння і в результаті простих підрахунків отримують шукане значення.

Популярним і водночас простим є алгоритм k-найближчих сусідів [9, p. 258]. При розв'язанні задач за допомогою цього алгоритму шукаються k найближчих точок до тієї, для якої робиться прогноз, і обраховується загальна кількість серед них, що відповідає певному класу. Результатом прогнозу буде значення, отримане для найбільшої кількісті точок-сусідів.

Наївний Баєс – це ймовірнісний класифікатор, який було створено на основі Баєсових теорем. Головною особливістю цього алгоритму є припущення про незалежність кожної ознаки об'єкту від інших [9, p. 347].

Метод опорних векторів – ще один алгоритм МН. Його широко використовують для вирішення задач класифікації, і не тільки. Головною ідеєю цього методу є побудова граничної лінії між класами таким чином, щоб бути якомога далі від кожної точки цих класів. Тоді прогнозування належності нового об'єкта якомусь класу проходить дуже швидко, зображенням його в тому ж просторі і залежно від того, з якої сторони лінії він буде розташований, визначатиметься і його клас [9, p. 202].

Логістична регресія – алгоритм МН для вирішення задач класифікації з бінарною цільовою змінною (вона може набувати тільки двох значень). Цей алгоритм використовує те саме рівняння, що і лінійна регресія, тільки, на відміну від неї, цільова змінна набуває не довільного числового значення, а одне з значень {0, 1}. Тому для прогнозування результату використовують логістичну функцію [9, p. 126].

Дерево рішень – ще один підхід до розв'язання задач МН. Як зрозуміло з назви, в основі цього алгоритму лежить побудова дерева, здатного виводити необхідний результат. Для задач класифікації це певний клас, якому належить об'єкт, а для задач регресії – число [9, p. 250].

Випадковий ліс – модифікація попереднього алгоритму, тільки замість побудови одного дерева тут будується декілька, а потім голосуванням визначається цільове значення. Тобто, кожне з побудованих дерев виводить певний результат, а для визначення найімовірнішого обирається той, що зустрічається найчастіше [9, p. 255].

Також для виконання класифікації певних об'єктів можна застосовувати нейронні мережі. Є декілька видів нейронних мереж, які мають різну архітектуру. Для задач класифікації використовують (у випадку навчання з учителем) згорткові нейронні мережі [6, p. 169] та перцептрон [6, p. 2].

Згорткові нейронні мережі, які також використовують для розв'язання задач класифікації, в основному спрямовані на роботу з зображеннями.

1.2. Робота з даними

У цій роботі вирішується задача класифікації міських звуків. А отже, дані представлені аудіофайлами. Для кращого розуміння, як працювати з ними, необхідно розібратися, що таке звук і які є особливості роботи з ним.

Звук подається у вигляді звукового сигналу, що має такі параметри, як частота, децибел тощо. Типовий аудіосигнал може бути виражений як функція амплітуди та часу (рис. 1.1).

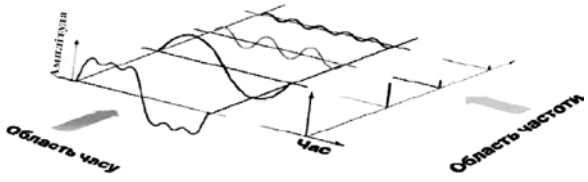

Рис. 1. Аудіосигнал як функція амплітуди та часу [4]

Програмно можна розпізнати такі формати аудіо, як, наприклад, MP3, WAV та WMA. Це означає, що комп'ютер може зчитувати та аналізувати файли такого формату як аудіозаписи.

Зображення форми звукового сигналу допомагає побачити зміну амплітуди звуку у часі.

Ще однією важливою характеристикою аудіосигналу є спектр його частот. Спектрограмою звуку є візуальне представлення спектра частот

звуку або інших сигналів, оскільки вони змінюються з часом. А отже, перша вісь являє собою частоту, а друга – час.

Для розв'язання поставленої задачі ми брали набір аудіофайлів із сайту Kaggle [11]. Там доступно два набори даних, один з яких має мітки (labels) класів, а інший – ні. Усі аудіофайли мають довжину запису чотири секунди або менше та представлені у форматі WAV.

Використовувати дані у такому вигляді неможливо. Кожний аудіозапис необхідно привести у вигляд, що буде доступний алгоритму МН.

У Python є засоби для роботи зі звуком, а саме: IPython.display.Audio – створює аудіооб'єкт, використовується для відображення елементів керування аудіо у веб-інтерфейсі [3]; Friture – графічна програма, розроблена для проведення частотно-часового аналізу аудіо в реальному часі; LibROSA – модуль, призначений для аналізу аудіозаписів та музики, який надає можливість працювати з багатьма найбільш популярними ознаками в музичному аналізі [5].

У мові R також є подібні засоби, наприклад, SEEWAVE – пакет, який дає змогу аналізувати та синтезувати аудіо.

1.3. Вибір та оптимізація моделей

Вибір моделі в кожному випадку залежить від особливостей задачі, яку треба вирішити. Також важливе значення має кількість класів, у випадку розв'язання задач класифікації. Їх може бути два і більше. Якщо їх лише два, то доцільним буде застосування алгоритму логістичної регресії. Але це не означає, що тільки він буде добре передбачати цільову змінну. Запорукою успіху вирішення певної задачі буде порівняння декількох моделей і їх точності у вирішенні конкретної задачі.

Неможливо заздалегідь передбачити, який саме алгоритм працюватиме ефективно на обраних даних, тому важливим є застосування декількох алгоритмів і їх порівняння між собою. До того ж, перевірка параметрів також має важливе значення і може підвищити точність прогнозу.

1.3.1. Grid-пошук

Grid-пошук (*англ.* Grid Search) – метод оцінки параметрів моделі, який використовують для безпосередньої її оптимізації. Для виявлення параметрів, які краще за інших працюють із цією моделлю, необхідно для кожного параметра вказати масив значень. Серед цих значень і шукатимуть оптимальне.

Важливим аспектом тут є те, що цей метод шукає найкращу *комбінацію* параметрів серед усіх доступних.

Наприклад, необхідно знайти оптимальну модель для розв'язання певної задачі. Використовують алгоритм «випадковий ліс». Щоб отримати необхідний результат, перевіряють деякі параметри, наприклад, кількість дерев і максимальну довжину дерева. Перший параметр задається як масив значень [3; 5; 7], а другий – [10; 20]. Тоді перевірка ефективності моделей буде здійснюватися для кожної пари з [(3, 10), (3, 20), (5, 10), (5, 20), (7, 10), (7, 20)].

Важливим зауваженням буде те, що ця перевірка не гарантує, що обрана в результаті модель буде найефективнішою. Вона буде найкращою серед перевірених, але не серед усіх можливих. Існує вірогідність, що можлива краща комбінація параметрів моделі, аніж обрана. Просто її не було розглянуто.

1.3.2. Крос-перевірка

При застосуванні алгоритмів МН для розв'язання певних задач виникає необхідність виміру точності прогнозу, оскільки це дає уявлення про ефективність обраної моделі.

Зрозуміло, що перевірка точності прогнозу на даних, які використовували для навчання моделі, є неможливою, оскільки в цьому випадку прогноз буде максимально точним, а поведінка моделі на нових даних буде непередбачуваною. Тому для оцінки ефективності моделей використовують такий підхід, як крос-перевірка (*англ.* Cross-Validation).

Існує два варіанти крос-перевірки: метод відкладених даних і контроль за k-блоками [1, с. 129].

У першому випадку тренувальні дані розбивають на два набори. Один використовується для навчання моделі, а інший – для перевірки її ефективності. Тобто, при використанні цього методу певна частина даних відкладається, а отже, стає контрольною (використовується для оцінки моделі). Варто зауважити, що розбиття набору на тренувальну і тестову вибірки відбувається випадковим чином, щоб запобігти можливості нерепрезентативної оцінки.

У другому випадку початковий набір розбивається на k блоків однакової довжини. Перевірка відбувається таким чином, що певна частина цих блоків використовується для навчання моделі, а інша – для оцінки передбачень. Зазвичай, їх розбиття проходить у такий спосіб, що для навчання береться (k-1) блок, а для передбачення –

той, що залишився. Сама перевірка відбувається таким чином, щоб кожний із блоків був використаний для передбачень. Тобто, якщо є k блоків, то перевірка відбувається k разів, а для передбачень береться і-й (i = 1, 2, ..., k) блок. Для навчання використовуються усі інші.

У результаті крос-перевірки за k-блоками доступна інформація про точність передбачення для кожної ітерації і середня оцінка ефективності моделі.

2. ВИКОРИСТАННЯ МН У ЗАДАЧІ КЛАСИФІКАЦІЇ МІСЬКИХ ЗВУКІВ

2.1. Алгоритмічне та програмне забезпечення

Для розв'язання задачі класифікації міських звуків ми обрали мову програмування Python (3.7.2) і використовували бібліотеки scikit-learn, keras, numpy, pandas та librosa.

Scikit-learn – потужна бібліотека [8], яку залучають з метою вирішення задач МН із використанням мови програмування Python. Вона містить велику кількість різноманітних алгоритмів МН, засобів обробки даних (наприклад, масштабування даних), засобів оцінки моделей МН та багато іншого.

Keras – відкрита бібліотека [2], яка забезпечує можливістю використання нейромереж. Вона є надбудовою над такими фреймворками, як TensorFlow та Theano [10].

Pandas [7] – Python-бібліотека для роботи з даними (їх аналізу та обробки). Забезпечує спеціальними структурами даних, такими як Series та DataFrame, і операціями для маніпулювання цими структурами.

Librosa – як уже зазначено вище, бібліотека для роботи зі звуком [3]. У цьому проекті використовують для витягування ознак (представлених числовими даними) з аудіо.

Алгоритмами МН, які використовують для класифікації, було обрано метод k-найближчих сусідів, «випадковий ліс», метод опорних векторів та нейронні мережі, які є методом вирішення задач глибокого навчання, що своєю чергою є підрозділом МН.

Експерименти проводили на комп'ютері Asus із процесором Intel Pentium N3710, 4 ядрами з частотою 1.60 гГц, графічною картою Intel HD Graphics for Intel Celeron Processor N3000 Series та оперативною пам'яттю 4 Гб. Операційна система – Linux Mint 19.1 Cinnamon.

2.2. Робота зі звуком

Для наочності застосування МН у задачах класифікації взято набір даних, які представлені аудіофайлами різних міських звуків із довжиною запису не більше ніж чотири секунди. Цей набір поділяється випадковим чином на два.

Перший із них містить 5435 аудіо, які призначені для тренування моделей МН. Кожний із цих аудіозаписів містить мітку (label).

Другий набір призначений для тестування моделей і містить 3297 записів. У цьому наборі немає міток, що моделює справжню роботу інженерів МН, бо на практиці модель навчається на відомих даних, а застосовується для нових, досі не бачених (а отже, даних без міток). Вся суть МН зводиться до ефективного підбору моделі і подальшого її застосування для прогнозу міток нових об'єктів.

Кожне аудіо з набору належить до одного з десяти класів: звук кондиціонера; сигнал автомобіля; звуки дітей, що граються; гавкіт собаки; буріння; звук двигуна на холостому ходу; звук пострілу; звук відбійного молотка; сирена; звуки вуличної музики.

2.2.1. Витягування ознак з аудіофайлів

Більшість алгоритмів МН вміють працювати лише з числовими даними, а отже, постає задача обробки аудіозаписів і витягування їхніх характеристик.

Першою і однією з найважливіших характеристик звуку є швидкість перетину нуля (zero crossing rate). Ознака характеризує швидкість зміни знаків вздовж сигналу – швидкість, з якою сигнал змінюється від позитивного до негативного і навпаки. Цю функцію використовують здебільшого для розпізнавання мовлення та повернення інформації про музику. Зазвичай значення параметра є більш високим для звуків, у яких ударна партія має велике значення, наприклад у рок-музиці.

Для отримання значення цієї характеристики використовується функція *feature.zero_crossing_rate(y)* з бібліотеки librosa, де y – аудіочасовий ряд, а результат її застосування – відповідний масив значень швидкості перетину нуля для цього ряду.

Звуковий спектр відображає різні частоти, які наявні у звуці. Для детальності аналізу звуків використовується низка характеристик, які певним чином пов'язані з ним.

Спектральний центроїд – наступна ознака звукового сигналу, яка вказує, де розташований «центр мас» для звуку. Обраховують як середньозважене значення частот, наявних у звуці.

Для отримання значення спектрального центроїда використовується функція *feature.spectral_centroid(y, sr)*, де y – аудіочасовий ряд, sr – частота дискретизації звуку y. Вона обчислює значення спектрального центроїда для кожного кадру у сигналі.

Наступною характеристикою, необхідною для вирішення задачі класифікації МЗ, є спектральний поворот, який представляє частоту, нижче якої заданий відсоток від загальної спектральної енергії. Функція *feature.spectral_rolloff(y, sr)* обчислює частоту обертання для кожного кадру в сигналі. Функція *feature.spectral_bandwidth(y, sr)* повертає смуги частот для кожного кадру.

Наступна характеристика є дуже цікавою і корисною для розв'язання задач, пов'язаних зі звуком або музикою. Вона являє собою увесь спектр, який проектується на 12 полу тонів октави і обраховується функцією *feature.chroma_stft(y, sr)*.

Під час роботи зі звуком також важливим є обчислення середньоквадратичного значення (RMS), що є можливим завдяки *feature.rms(y)*, яка обраховує середньоквадратичне значення для кожного кадру.

При аналізі звуку також можливим є використання тональних характеристик центроїда, які обраховуються функцією *feature.tonnetz(y, sr)*.

Мел-частотні кеспральні коефіцієнти (Mel frequency cepstral coefficients) – остання характеристика, яка потрібна для виконання класифікації міських звуків. І складається вона з невеликого набору ознак, які описують загальну форму спектральної огинаючої. Для отримання цих ознак необхідно скористатися функцією *feature.mfcc(y, sr)*.

2.2.2. Аналіз та обробка даних

Коли кожний звук представлений рядом ознак, необхідно зробити ряд перетворень для використання їх моделями МН.

У першу чергу, для зручності є доцільним представлення даних як DataFrame, який є форматом даних серед представлених у бібліотеці pandas. Тепер усі дані відображаються у вигляді табл. 1.

Таблиця 1. **Вигляд DataFrame**

	zero_crossing_rate	spectral_centroid	spectral_bandwidth	spectral_rolloff	chroma_stft	rmse	tonnetz_fifth_x	tonnetz_fifth_y	tonnetz_minor_x
0	-0.358866	-0.337109	-0.170332	-0.367131	1.042656	0.328102	-0.080330	0.330806	-0.487840
1	-0.287618	-0.109106	0.109364	0.077577	-0.713026	1.548286	0.094035	0.198044	-0.890504
2	2.576966	2.296752	1.327939	2.023059	0.503646	-0.059925	0.303920	-0.555116	0.392151
3	-0.112072	-0.192276	-0.774478	-0.295867	-2.194357	1.104959	0.079355	-1.843641	-1.681755
4	-1.024086	-0.975340	-0.203525	-0.883009	0.131579	0.764461	0.361874	0.240133	0.490246

Отже, табл. 1 має набір рядків, кожний з яких відповідає певному аудіозапису. Кожний рядок має *id* – порядковий номер аудіозапису (і за сумісництвом, назва аудіофайлу), якому відповідають певні характеристики, а саме ознаки звуку, які було описано в попередньому підпункті, та клас, до якого належить звук (для тренувальної вибірки).

Під час підготовки даних до використання їх алгоритмами МН слід пересвідчитись, що немає «відсутніх даних», тобто всі комірки таблиці мають значення. При перевірці тренувального і тестового наборів таких даних не було виявлено, а отже, витягування ознак аудіо пройшло правильно.

Для підвищення ефективності навчання необхідним кроком є масштабування даних. Алгоритми МН краще працюють на нормалізованих даних, тобто даних, які були попередньо оброблені з метою їх підлаштування під єдину числову шкалу. Залежно від діапазону значень ознак враховується їхня важливість щодо інших. Якщо одна ознака має діапазон значень від 0 до 10, а інша – від 0 до 1, то перша ознака враховуватиметься алгоритмом МН, як більш важлива (в 10 разів), ніж друга [1, с. 78].

Для масштабування даних у бібліотеці scikit-learn є така функція, як *preproccessing.StandardScaler()*. Тому все, що необхідно зробити – це застосувати її до характеристик звуку.

Також більшість алгоритмів МН не можуть працювати з мітками, представленими у форматі String, – тільки з числовими значеннями. Отже, наступним кроком обробки даних для подальшого навчання на них є перетворення міток класів на числові позначення. У випадку бінарної

класифікації зазвичай класам надаються значення 0 та 1.

Для класифікації міських звуків було визначено 10 різних класів. Оскільки мітки цих класів не є числовими значеннями, постає задача їх перетворення у правильний формат. Для цього використовується *preprocessing.LabelEncoder()* з бібліотеки scikit-learn.

Тепер усі класи представлені числовими значеннями відповідно: 0 – air_conditioner, 1 – car_horn, 2 – children_playing, 3 – dog_bark, 4 – drilling, 5 – engine_idling, 6 – gun_shot, 7 – jackhammer, 8 – siren, 9 – street_music.

Також варто зауважити, що кількість екземплярів кожного класу є різною. Більшість класів містить приблизно 600 об'єктів, тільки два з них – значно менше (в класі «постріл пістолета» близько 200, а в «сигнал машини» – 300).

2.3. Класифікація

2.3.1. Вибір та оптимізація моделей МН

Було обрано такі алгоритми: метод k-найближчих сусідів, метод опорних векторів, «випадковий ліс» і багатошаровий перцептрон.

Тепер необхідно визначити, які параметри для кожної моделі дають найкращий результат. Для цього використовуємо метод grid-пошуку.

```
def classify_with_grid_search(model, parameters, dataset):
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    gsearch = GridSearchCV(model, parameters, cv=5)
    gsearch.fit(X, y)

    return gsearch.best_estimator_
```

Рис. 2. Функція пошуку найкращої моделі

Функція *classify_with_grid_search(model, parameters, dataset)* приймає на вхід такі параметри, як model – алгоритм МН, для якого знаходять найкращі параметри (оціночна функція), parameters – словник з іменами параметрів (ключі) і списком налаштувань (значення) та dataset – набір даних, на яких навчаються моделі. Спочатку набір даних розбивається на масив ознак звуків та масив міток (тобто класів) звуків. Потім визначається екземпляр класу сіткового пошуку, який використовує задану оціночну функцію, параметри та крос-перевірку з розбиттям на 5 блоків. Наступним етапом є його навчання на визначених даних. Результат виконання функції – оцінщик, який дав найкращий результат.

Отже, після застосування цієї функції до кожного алгоритму визначено найкращі параметри для передбачень на доступних даних.

У випадку алгоритму «випадковий ліс» найкращі параметри шукали серед таких: для n_estimators – кількості дерев прийняття рішень – задавали значення 2, 5, 10, 15; критерій пошуку criterion шукали серед gini та entropy, вага класів – серед None , balanced та balanced_subsample, а для максимальної глибини дерева задавали значення 15, 20, 21, 23, 25. У результаті було визначено, що найефективніша модель має такі параметри: вага класів – balanced, критерій пошуку – gini, максимальна глибина дерев – 20, кількість дерев – 15.

Для методу опорних векторів найкращими параметрами виявилися: C – 15, ядро – rbf, вага класів – None.

У випадку з методом k-найближчих сусідів це: кількість сусідів – 5, ваги – залежать від відстані (distance), метрика – minkovski.

Для багатошарового перцептрона (Multi-Layer Perceptron) найкращими параметрами є: активаційна функція – tanh, значення альфа – 0.1, максимальна кількість ітерацій – 500.

Отже, в результаті виконання функції *classify_with_grid_search(model, parameters, dataset)* було отримано найкращі серед заданих параметрів для навчання моделей на доступних даних.

2.3.2. Етап навчання

Зараз, коли відомі найоптимальніші параметри для кожної моделі, слід їх перевизначити на знайдені.

Наступним етапом є навчання моделей на тренувальному наборі даних. Але перед цим слід оцінити ефективність передбачень кожної з них, щоб обрати найкращу для класифікації міських звуків. Для цього скористаємося функцією *evaluate(model, dataset)*, в якій для оцінки кожної з моделей застосовується метод крос-перевірки.

```
def evaluate(model, dataset):
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    scores = cross_val_score(model, X, y, cv=5)

    return scores
```

Рис. 3. Функція оцінки моделі за допомогою крос-перевірки

У тілі функції спочатку набір даних поділяють на два: перший використовують для навчання моделі, а другий – для оцінки зроблених

Таблиця 2. **Оцінка точності передбачень методів**

Модель	1 – ітерація	2 – ітерація	3 – ітерація	4 – ітерація	5 – ітерація	середнє
«Випадковий ліс»	0.87511478	0.87040441	0.87120515	0.8839779	0.86359447	0.872859342
Опорних векторів	0.93847567	0.93474265	0.92916283	0.9373849	0.93640553	0.935234316
k-найближчих сусідів	0.92470156	0.90441176	0.90524379	0.9106814	0.90506912	0.91
Багатошаровий перцептрон	0.92653811	0.90349265	0.90239411	0.89769585	0.89769585	0.906336932

прогнозів зіставленням реальних класів із передбаченими. Потім застосовують метод оцінки за допомогою крос-перевірки, зберігаючи результати передбачення для кожної ітерації. До того ж, у цьому випадку застосовують розбиття набору даних на 5 блоків. Результатом виконання функції буде масив оцінок точності передбачень.

Отже, кожна з моделей (із визначеними параметрами) проходить перевірку методом розбиття тренувальної вибірки на k блоків.

У результаті виконання перевірки отримано оцінки точності (табл. 2).

Бачимо, що найгірший результат серед отриманих у «випадковому лісі». В середньому, точність його передбачень становить 87 %. Непогані оцінки дають метод k-найближчих сусідів і багатошаровий перцептрон. А найкращим алгоритмом для класифікації даних, що використовується для розв'язання поставленої задачі, є метод опорних векторів, оскільки точність його прогнозів в середньому дає приблизно 93 %.

Отже, доцільним є використання останнього алгоритму для виконання прогнозів на невідомих раніше даних.

2.3.3. Передбачення

Задля розуміння, як передбачення працюватимуть на нових даних, необхідним кроком є перевірка їх точності на даних, у яких відомі мітки класів.

Отже, тренувальна вибірка розбивається на два набори даних задля створення тренувальної і тестової вибірок. Це робиться за допомогою спеціальної функції, яка називається *train_test_split*, вбудованої в бібліотеку Scikit-learn. Як параметри функції задаються набір даних, на яких навчається модель, набір міток, що йому відповідає, розмір тестової вибірки (значення задається десятковим числом від 0 до 1, що відповідає проценту даних, що беруться з оригінальної вибірки для утворення тестової, решта залишається в тренувальному наборі даних; найчастіше беруться значення 0.33 та 0.2) і значення, яке використовується генератором випадкових чисел як початкове число.

У результаті розбиття отримано тренувальну вибірку довжиною в 4348 рядків і тестову з довжиною 1087. Розбиття проходило за допомогою генератора випадкових чисел, що відкидає варіант нерепрезентативного розподілу даних.

Наступним кроком є навчання обраних моделей на новоутвореному тренувальному наборі даних та застосування їх для класифікації об'єктів тестової вибірки.

Після виконання попереднього кроку застосовуємо функцію *accuracy_score(y_test, y_pred)*, яка оцінює точність класифікації, зробленої моделлю, з реальними даними. У результаті перевірки ефективності моделей отримаємо такі результати: використання методу опорних векторів для передбачення класів міських звуків з тестового набору даних дало приблизно 92 % правильних відповідей, а використання багатошарового перцептрона – 89 %. Якщо порівнювати ці результати з середніми оцінками крос-перевірки, то вони є трохи гіршими, приблизно на 1 %.

```
def comparison(y_real, y_pred):
    df = pd.DataFrame({'real': y_real, 'pred': y_pred})
    df = df[~(df.real == df.pred)]
    df = df.groupby(['real','pred']).size().reset_index().rename(columns={0:'count'})
    maximum = df.loc[df['count'].idxmax()]

    return maximum
```

Рис. 4. Функція для знаходження помилок, що зустрічаються найчастіше

Для аналізу помилок, які були зроблені під час класифікації, необхідно порівняти, як часто кожне справжнє значення плутається з передбаченим. Це необхідно для того, щоб розуміти, де найвірогідніше буде зроблено помилку в майбутньому. Отже, скористаємось функцією, яка рахує кількість помилок для кожної пари значень реальне число/передбачене і виводить інформацію про них.

У результаті застосування функції помічено, що моделям «випадковий ліс» та «метод опорних векторів» характерно класифікувати об'єкти з міткою «вулична музика» (9) як «гру дітей» (2). А модель багатошарового перцептрона помиляється в класифікації «гавкоту собаки» (3) і також розпізнає його як «гру дітей» (2).

Вищевикладена інформація може бути корисною за подальшої класифікації, тільки вже даних, які не мають завчасно визначених міток класів.

2.4. Аналіз результатів

Для чотирьох обраних алгоритмів, а саме k-найближчих сусідів, опорних векторів, випадкового лісу і багатошарового перцептрона, застосовано метод grid-пошуку, який знайшов найкращі серед визначених параметри для кожного з них. Після чого усі моделі пройшли перевірку ефективності, тобто точності її передбачень, за допомогою контролю за k-блоками. У результаті знайдено модель, яка дає найбільшу точність передбачень. Це метод опорних векторів. Точність прогнозу в середньому становить 93 %.

Наступним етапом є класифікація та її оцінка. Навчання проводили на одному наборі даних, а на іншому відбувалося прогнозування класів. У результаті виявилося, що точність роботи моделей є гіршою від попередньої оцінки, приблизно на 1 %.

2.5. Використання нейронних мереж для класифікації

У Scikit-learn є можливість використання нейронних мереж для вирішення поставлених задач. Прикладом може слугувати MLPClassificator(), який застосовано під час знаходження ефективних моделей МН для класифікації МЗ. Точність передбачень не є найкращою, а отже треба шукати інші шляхи покращення ефективності розпізнавання міських звуків.

Для роботи з нейромережами краще використовувати спеціальні бібліотеки, які призначені виключно для цього. Однією з таких бібліотек є Keras, яка за замовченням використовує TensorFlow – потужний фреймворк для створення і застосування нейронних мереж.

Тому важливим етапом для знаходження оптимального варіанту розв'язання задачі класифікації МЗ є створення нейромережі і її застосування для вирішення цієї проблеми.

Для побудови нейронної мережі використовують модель Sequential – послідовну модель. Для побудови прихованих шарів та вихідного (суматору) – Dense також з бібліотеки Keras. Для регуляризації моделі – Dropout. А для оптимізації – SGD. Бібліотека pandas забезпечує роботу з даними. Кодування міток – OneHotEncoder з бібліотеки scikit-learn.

Спочатку необхідно завантажити дані для роботи з ними. В цьому випадку створено два набори даних: тренувальний і тестовий.

Щоб модель могла використовувати дані для навчання, необхідно розбити їх на набір ознак та мітки класів.

Оскільки на тестовій вибірці, яка не має міток класів, неможливо оцінити точність роботи моделі, необхідним кроком є створення набору даних для цього. Щоб мати можливість оцінки ефективності прогнозів нейромережі, поділимо тренувальну вибірку на дві так само, як і у попередньому пункті, де розглядалися інші моделі МН для розв'язання поставленої задачі. Розмір новоутвореної вибірки становить 20 % від усієї, а решта даних зберігається для навчання моделі.

Необхідно зазначити, що під час роботи, коли в нас є багато класів, виникають деякі проблеми, зокрема неспроможність нейронної мережі використовувати класи в початковому вигляді. Отже, необхідно виконати їх кодування для коректної роботи моделі нейронної мережі. Для кодування класів використовується функція OneHotEncoder, що представлена в пакеті sklearn.preproccessing.

Нарешті, можна будувати нейронну мережу. Як уже зазначено, для цього використовують послідовну модель.

Кількість прихованих шарів визначається трьома. Як функцію активації береться функція *relu* – вона для кожного елементу повертає max(x, 0). Перший шар має розмірність вихідного простору 112. Також тут визначається розмірність вхідного набору даних, тобто кількість ознак, за якими проводиться навчання. У нашому випадку це 32 ознаки звуку. Розмірність другого прихованого шару дорівнює 84, а третього – 48.

Після роботи кожного з прихованих шарів застосовується регуляризація. Вона проводиться за допомогою методу відсікання частини даних, що

означає ігнорування певної кількості випадково обраних нейронів під час навчання. Регуляризація призводить до зменшення чутливості нейронної мережі до конкретних ваг нейронів. В результаті модель стає здатною до кращого узагальнення і зменшується ймовірність її перенавчання. Значення 0.1 визначає долю нейронів, які будуть ігноруватися під час навчання.

Останній шар визначає вихідні дані, у випадку класифікації це клас об'єкта. Його активаційною функцією є *softmax*, а розмірність простору визначається кількістю можливих класів.

Після створення моделі нейронної мережі потрібне її налаштування для навчання, а саме визначення показника для оцінки моделі під час навчання та тестування, оптимізатора та функції втрат.

Спочатку обираємо оптимізатор та визначаємо його параметри (в іншому випадку використовуються значення за замовчуванням). Отже, для оптимізації моделі обрано стохастичний градієнтний спуск (SGD) із застосуванням імпульсу Нестерова – прискоренням градієнта. Такі налаштування зазвичай використовуються для неглибоких нейронних мереж.

Після визначення усіх параметрів оптимізатора відбувається налаштування моделі. Функцією втрат визначено *categorical_crossentropy*, оскільки дані, представлені у цільовому наборі, є категоріальними. Також вказується оптимізатор, визначений перед цим. Останнім уточнюється показник, що оцінюється моделлю. Зазвичай, і цей проект не є винятком, це точність.

Наступним етапом є навчання моделі на тренувальних даних. При цьому визначається кількість ітерацій (epochs) для навчання нейронної мережі та кількість зразків на оновлення градієнту (batch_size). Параметри визначено числами 300 і 128 відповідно.

Після запуску нейронної мережі отримано такий результат: 92,82 % – точність роботи моделі.

Також виконано класифікацію даних, які не містять міток класів.

Висновки

У цій роботі розглянуто особливості використання МН у задачах класифікації на прикладі класифікації міських звуків, розроблено чотири моделі МН для розв'язання поставленої задачі. Створені моделі дають змогу ідентифікувати міські звуки з різною природою походження.

Отримано такі результати:

1. Проведено перетворення звуків на сукупність числових ознак із подальшою їх обробкою для оптимізації ефективності роботи моделей МН.

2. Розроблено чотири моделі МН із використанням алгоритмів k-найближчих сусідів, методу опорних векторів, багатошарового перцептрона та випадкового лісу для розв'язання поставленої задачі.

3. Розроблено послідовну модель нейронної мережі для вирішення задачі класифікації міських звуків.

4. Застосовано моделі для передбачення класів об'єктів із тестової вибірки.

5. Проаналізовано роботу кожної моделі та точність її передбачень.

При перевірці роботи моделей на тестових даних отримано такі результати: для k-найближчих сусідів оцінка точності становить 90 %, для методу опорних векторів – 92 %, для «випадкового лісу» – 88 %, а для багатошарового перцептрона – 90 %. При розробці послідовної моделі нейронної мережі з трьома прихованими шарами отримано 93 % правильно передбачених цільових значень – класів.

Проаналізувавши ефективності роботи вищезгаданих моделей МН, можна зробити висновок, що застосування методу опорних векторів та послідовної моделі нейронної мережі є найкращими серед розглянутих для розв'язання поставленої задачі – класифікації міських звуків.

Список літератури

1. Бринк Х. Машинное обучение / Х. Бринк, Д. Ричардс, М. Феверолф. – Санкт-Петербург : Питер, 2018. – 336 с. – (Библиотека программиста).
2. Keras: The Python Deep Learning library [Electronic resource]. – Mode of access: https://keras.io. – Title from the screen.
3. Module: display [Electronic resource]. – Mode of access: https://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html. – Title from the screen.
4. Music Genre Classification With Python [Electronic resource]. – Mode of access: https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8. – Title from the screen.
5. Music software written in Python [Electronic resource]. – Mode of access: https://wiki.python.org/moin/PythonInMusic. – Title from the screen.
6. Nielsen M. Neural Networks and Deep Learning / Michael Nielsen. – Determination Press, 2015. – 216 p.
7. Python Data Analysis Library [Electronic resource]. – Mode of access: https://pandas.pydata.org. – Title from the screen.
8. scikit-learn [Electronic resource]. – Mode of access: https://scikit-learn.org/stable/. – Title from the screen.
9. Shalev-Shwartz S. Understanding Machine Learning: From Theory to Algorithms / S. Shalev-Shwartz, S. Ben-David. – Cambrige University Press, 2014. – 449 p.
10. This Is What Makes Keras Different, According To Its Author [Electronic resource] // Forbes. – 2016. – Mode of access: https://www.forbes.com/sites/quora/2016/08/25/this-is-what-makes-keras-different-according-to-its-author/#2c7851fa66cf. – Title from the screen.
11. Urban Sound Classification [Electronic resource]. – Mode of access: https://www.kaggle.com/pavansanagapati/urban-sound-classification. – Title from the screen.

References

Brink, H., Richards, D., & Feverolf, M. (2018). *Mashinnoe obuchenie.* Sankt-Peterburg: Piter [in Russian].

Keras: The Python Deep Learning library. Retrieved from https://keras.io/.

Module: Display. Retrieved from https://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html.

Music Genre Classification With Python. Retrieved from https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8.

Music software written in Python. Retrieved from https://wiki.python.org/moin/PythonInMusic.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning.* Determination Press.

Python Data Analysis Library. Retrieved from https://pandas.pydata.org/.

scikit-learn. Retrieved from https://scikit-learn.org/stable/.

Shalev-Shwartz, S., & Ben-David, S. (2017). *Understanding machine learning: From theory to algorithms.* Cambridge: Cambridge University Press.

This Is What Makes Keras Different, According To Its Author. Forbes, 2016. Retrieved from https://www.forbes.com/sites/quora/2016/08/25/this-is-what-makes-keras-different-according-to-its-author/#2c7851fa66cf.

Urban Sound Classification. Retrieved from https://www.kaggle.com/pavansanagapati/urban-sound-classification.

M. Glybovets, A. Zhyrkova

USING MACHINE LEARNING IN SOUND CLASSIFICATION TASKS

The article describes usaging circumstances of Machine Learning (ML) methods for classifying auditory information . The problem was studied by using an example of urban sounds classification.

Machine Learning is a great possibility to process data presented differently, including text, images, and sounds. Natural language processing requires text analysis. Pattern recognition involves interaction with images. And, naturally, many ML tasks require working with numbers, e.g. prediction of the price for houses, or benign or malignant tumor recognition based on numerical characteristics of the tumor. As for sounds, most of researche is about music recognition and, in general, the related stuff. So, sound processing is a relevant topic for ML problems.

Environmental sounds classification is a field which attracts specialists from various spheres. Multimedia sensor networks and big variety of multimedia content, which describes urban scenes, is a huge part of it.

Urban sounds classification is an important problem which needs to be solved. Many researchers focus on locations (parks, streets, etc.) instead of sources (car horn, shoot, etc.). So, urban sounds classification is a reasonable, urgent problem.

Development of best-fit Machine Learning models for correct classification of urban sounds requires dealing with the following problems: sound representation as a set of numerical data considering its features for a possibility of ML algorithms to work with sound; data processing for optimization of ML models efficiency; using methods of ML models optimization for presenting the best parameters of each model; train models on training data; prediction of urban sound classes and checking the accuracy of these predictions on test data; neural networks development for solving the urban sounds classification problem.

Keywords: classification, machine learning, k-nearest neighbor, support vector machine, random forest, neural networks, k-fold cross-validation, holdout method, grid search.

Матеріал надійшов 31.05.2019