math_2020_079

Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича

Факультет математики та інформатики
(повна назва інституту/факультету)

Кафедра математичного моделювання
(повна назва кафедри)

Застосування методів виявлення ознак для машинного навчання засобами мови Python

Дипломна робота
Рівень вищої освіти - другий (магістерський)

Виконала:
студентка 6 курсу, групи 607
спеціальності 124 – Системний аналіз
(назва спеціальності)
Голик Дарія Юріївна
(прізвище, ім'я та по-батькові)
Керівник к.ф.-м.н, доцент кафедри
математичного моделювання Юрченко І.В.
(науковий ступінь, вчене звання, прізвище та ініціали)

До захисту допущено:
Протокол засідання кафедри № 7
від „15"грудня 2020 р.
зав. кафедри_____ проф. Черевко І.М.

Чернівці – 2020

Анотація

У даній роботі застосовано методи HOG та опорних векторів (з використанням бібліотеки Scikit-learn) для побудови програмної реалізації алгоритму виявлення обличчя на зображенні засобами мови Python.

Список символів та позначень

• **ML**	Машинне навчання
• **RF**	Випадковий ліс
• **AI**	Штучний інтелект
• **SVM**	Метод опорних векторів
• **SL**	Навчання з учителем
• **USL**	Навчання без учителя
• **HOG**	Гістограма направлених градієнтів
• **LBP**	Локальні бінарні шаблони
• **SURF**	Прискорені стійкі признаки
• **KNN**	k – найближчих сусідів
• **NB**	Наївний Байєс

Зміст

Вступ

На етапах постановки задачі машинного навчання і формування даних не завжди зрозуміло, які ознаки важливі для побудови оптимального алгоритму, тому часто в даних зустрічається багато надлишкової інформації – шуму. Поява шумових ознак погіршує якість роботи алгоритму і уповільнює його роботу. Тому в більшості випадків перед вирішенням завдання класифікації, регресії або прогнозування необхідно вибрати ті ознаки, які є найбільш інформативними. Правильний вибір ознак може бути більш важливим завданням, ніж зменшення часу обробки даних, або поліпшення точності класифікації. Наприклад, в медицині, знаходження мінімального набору ознак, який є оптимальним для задачі класифікації, може бути корисним для розробки діагностичного тесту. Виявлення важливих ознак (наприклад, відбір генів, що відповідають певному типу раку) може допомогти розшифрувати механізми, що лежать в основі проблеми, яка представляє інтерес для дослідження.

Алгоритм виявлення ознак є основним інструментом у задачі розпізнавання образів. У наш час розпізнавання образів активно розвивається у криміналістиці (встановлення особи зловмисника з відео, отриманого з камер спостереження), медицині (вивчення знімків серця у розрізі, отриманих з МРТ, для діагностики серцевих захворювань), дорожньому русі (визначення марок машин, номерних знаків, встановлення особи водія, дотримання правил ДР), класифікації документів, розпізнаванні штрих кодів, розпізнаванні мови тощо. У Китаї такі системи розпізнавання працюють або проходять тестування уже кілька останніх років.

Задача розпізнавання образів має дві головні підзадачі: виявлення ознак та класифікація об’єктів. Найбільш поширеними алгоритмами, які використовуються для виявлення ознак з відео чи фото даних є гістограма направлених градієнтів (HOG), локальні бінарні шаблони (LBP), прискорені

стійкі признаки (SURF). Найкращими класифікаторами для таких задач є метод опорних векторів (SVM), випадковий ліс (RF), k – найближчих сусідів (KNN).

У даній роботі наведено історію розвитку машинного навчання; описано покрокову реалізацію алгоритму виявлення ознак HOG; розглянуто побудову класифікаторів для задач машинного навчання; засобами мови Python з використанням бібліотеки Scikit-Learn розроблено покроковий алгоритм розпізнавання образів з використанням методу виявлення ознак HOG та класифікатора SVM; проведено порівняння роботи SVM з наївним Баєсовим класифікатором (NB).

Розділ І. Історія розвитку та задачі машинного навчання

1.1. Визначення процесу машинного навчання

Машинне навчання (ML) являє собою область штучного інтелекту, яка використовує статистичні методи, щоб дати можливість комп'ютерним систем «вчитися» (наприклад, поступово покращувати продуктивність по конкретному завданні) з даних, не будучи явно запрограмованою [1].

З 1959 року Артур Семюель почав досліджувати вивчення і побудову алгоритмів, які можуть вчитися і робити прогнози на даних. З допомогою ML алгоритмів та даних стало можливим робити прогнози шляхом побудови моделі. Машинне навчання використовується в ряді обчислювальних задач, де проектування і програмування явних алгоритмів з хорошою продуктивністю є важким або неможливим. Приклади застосування ML алгоритмів включають фільтрацію електронної пошти, виявлення мережевих зловмисників та розпізнавання зображень. Останній приклад якраз і є об’єктом дослідження ML алгоритмів у даній магістерській роботі.

ML тісно пов'язане з обчислювальною статистикою, яка також фокусується на прогнозуванні з використанням комп'ютерів. ML має міцні зв'язки з математичною оптимізацією, яка надає методи та теорії у сфері застосувань. ML іноді поєднується з інтелектуальним аналізом даних, в якому останній – більше фокусується на аналізі розвідувальних даних і відоме як неконтрольоване навчання.

В області аналітики даних ML – це метод, який використовується для розробки складних моделей і алгоритмів, які піддаються прогнозуванню; в комерційному використанні це називається прогностичною аналітикою. Ці аналітичні моделі дозволяють дослідникам, інженерам і аналітикам «створювати надійні, повторювані рішення і результати» і розкривати «приховані ідеї» за допомогою вивчення історичних взаємозв’язків і тенденцій в даних.

Том М. Мітчелл представив більш формальне визначення алгоритмів, що вивчаються в області машинного навчання: «Кажуть, що комп'ютерна програма вчиться на досвіді E по відношенню до деякого класу задач T і показником продуктивності P, якщо його продуктивність при виконанні завдань в T, виміряна P, поліпшується з досвідом E.»

Таке означення йде слідом за пропозицією Алана Тьюринга в його статті «Обчислювальна техніка та інтелект», в якій питання «Чи можуть машини думати?» замінюється питанням: «Чи можуть машини робити те, що ми (як мислячі індивідууми) можемо робити?». У статті Тьюринга розкриваються різні характеристики, якими може володіти мисляча машина, і різні наслідки її функціонування [2].

1.2. Класифікація задач машинного навчання

Задачі ML зазвичай поділяються на кілька категорій [1]:

- Контрольоване навчання: комп'ютеру представлено зразкові вхідні дані і їх бажані результати, заданими «учителем». Мета полягає в тому, щоб вивчити загальне правило, яке відображає вхідні дані для виходів. В якості особливих випадків вхідний сигнал може бути тільки частково доступний або обмежений спеціально зворотним зв'язком.
- Напів-контрольоване навчання: комп'ютеру надається тільки неповний навчальний сигнал: тренувальний набір з деякими (часто багатьма) цільовими виходами відсутній.
- Активне навчання: комп'ютер може отримувати тільки навчальні ярлики для обмеженого набору примірників (на основі бюджету), а також повинен оптимізувати свій вибір об'єктів для отримання міток. При використанні в інтерактивному режимі вони можуть бути представлені користувачеві для маркування.

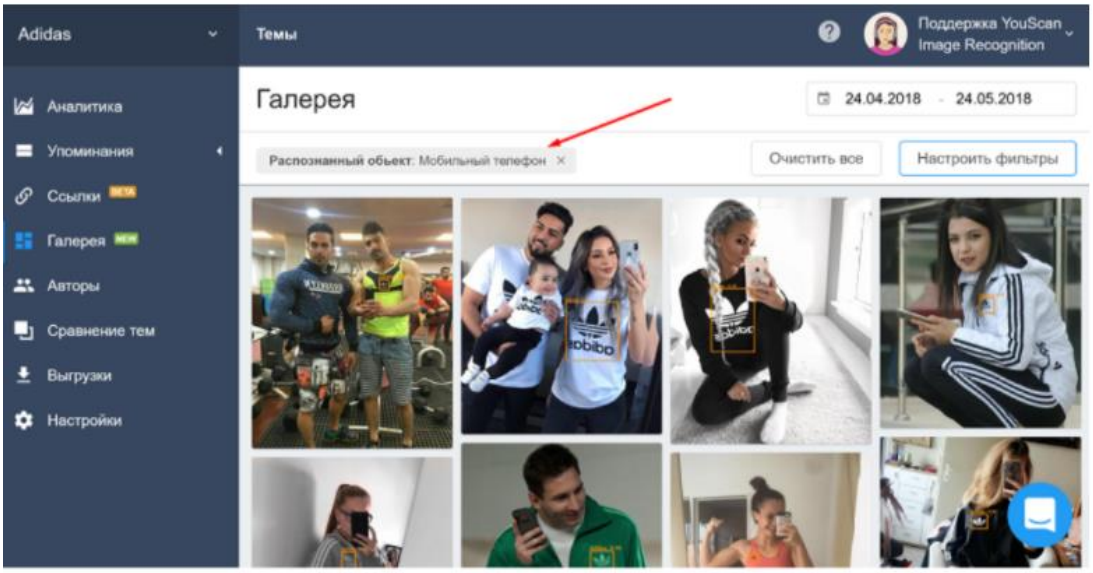
❖ Неконтрольоване навчання: ніякі мітки не доступні алгоритму навчання, залишаючи його самостійно, щоб знайти структуру у вході. Неконтрольоване навчання може бути самоціллю (виявлення прихованих шаблонів в даних) або засобом досягнення мети (навчання об'єктам).

❖ Посилене навчання: дані (у вигляді винагород і покарань) даються тільки як зворотний зв'язок з діями програми в динамічному середовищі, наприклад, водіння транспортного засобу або гра проти супротивника.

1.3. Сфери застосування алгоритмів виявлення ознак

- ***Маркетинг***

  Розпізнавання образів – перспективний напрямок в рекламі і маркетингу. Неймережі дозволяють за лічені години дізнатися речі, для пошуку яких в інших випадках потрібна велика команда професіоналів і тижні, а то й місяці досліджень. Наприклад, сервіс YouScan, система моніторингу соціальних мереж, відстежує згадку брендів в соцмережах. Причому робить це не тільки в тексті постів, але і на фотографіях, а також допомагає зробити певні висновки про продукт. За допомогою розпізнавання образів на фото знайшли цікаву закономірність, яка б нікому не прийшла в голову: серед тварин коти частіше зустрічаються з технікою Apple, а собаки – з брендом Adidas. Ця незвичайна інформація може стати в нагоді для удосконалення реклами [3].



При пошуку по логотипу Adidas сервіс YouScan відфільтрував фотографії зі смартфонами в руках власників

- ***Відеоспостереження***

  Розпізнавання образів на камерах міського відеоспостереження – це, мабуть, сама невідворотна перспектива використання машинного зору. У даний час в Україні тестується система розумного відеоспостереження з метою ідентифікації злочинців в місцях масового скупчення людей. До міської мережі камер підключена технологія від компанії NTechLab, яка вже допомогла затримати кілька десятків правопорушників. У Китаї подібна система відеоспостереження здатна розпізнавати не тільки особи, але і марки автомобілів та одягу на людях, що може бути згодом використано маркетологами для своїх досліджень.

Реальна робота розпізнавання образів та облич SenseTime у Китаї

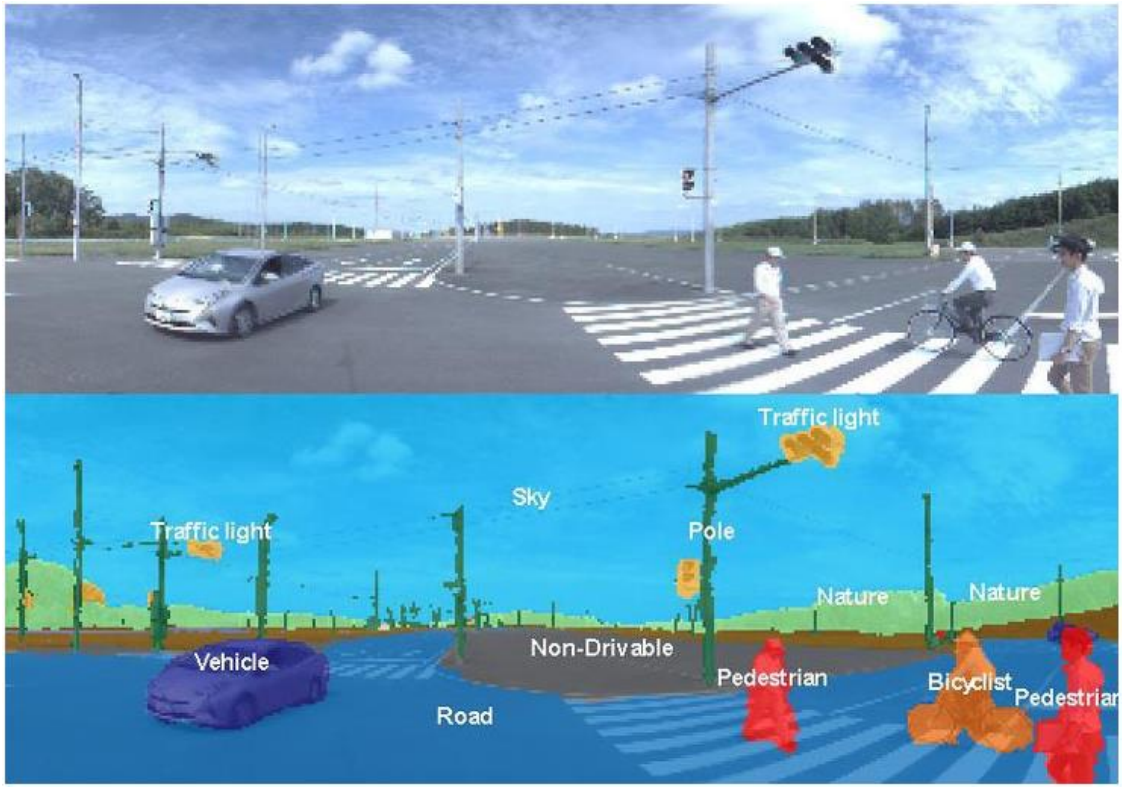
- ***Медицина***

  Розпізнавання образів вже стало справжнім проривом в медицині – у багатьох випадках комп'ютери помічають речі, які пропускають навіть найдосвідченіші лікарі. Вони виступають своєрідними помічниками, чия «технічна» думка підтверджує гіпотезу лікаря або дає привід для більш глибоких досліджень. В Україні ведуться розробки програмних комплексів для діагностики ракових утворень на знімках КТ, МРТ і ПЕТ. Для цього через нейромережу проганяють тисячі розмічених знімків, після чого точність розпізнавання нових знімків зростає до 95-97%. Створена нейромережа Google Inception аналізує мікроскопічне дослідження біопсії лімфатичних вузлів в пошуку ракових клітин в молочних залозах. Для людини це дуже довгий і трудомісткий процес, в ході якого легко помилитися або пропустити щось важливе, так як в деяких випадках розмір зображення становить 100 000 х 100 000 пікселів. Нейромережа Inception забезпечує чутливість близько 92% проти 72% у лікаря. Нейромережа не упустить з уваги всі підозрілі

ділянки знімків, хоча і допускаються помилкові опрацювання, які пізніше відфільтрує лікар.

- ***Автомобілі***

  Розпізнавання образів в автомобілях – це необхідна частина систем безпеки ADAS (Advanced driver-assistance systems). ADAS можуть бути реалізовані як складними засобами, на зразок радара та інфрачервоних датчиків, так і за допомогою монокулярної камери. Відеокамери цілком достатньо для того, щоб автомобіль в реальному часі зміг розпізнати пішоходів, знаки та світлофори. Однак таке розпізнавання «на льоту» – дуже ресурсномістке завдання, для виконання якого потрібен спеціалізований процесор. Toshiba вже протягом декількох років розвиває серію таких процесорів. Вони будують тривимірну модель на основі рухомого зображення з однієї камери, і тим самим помічають невідомі перешкоди на дорозі. Адже якщо нейромережа навчена розпізнавати тільки людей, розмітку та знаки, тоді те, що лежить на асфальті (покришка або шматок захисного засобу) не буде розпізнане і розцінене, як небезпека.



Процесори Visconti виділяють на зображенні зони, класифікують їх і допомагають автопілоту або ADAS прийняти рішення

- ***Дрони***

  Дрони з розпізнаванням зображень використовуються у багатьох напрямках життєдіяльності. Наприклад, норвезька компанія eSmart Systems розробила інтелектуальні рішення для енергомереж. В рамках одного з їхніх проектів – Connected Drone – дрони використовуються для пошуку несправностей на лініях електропередач. Навчені розпізнаванню елементів енергомереж, вони перевіряють цілісність проводів, ізоляторів та інших частин ЛЕП. Це особливо важливо для швидкої локалізації несправності, коли від лінії залежить електропостачання міста або підприємства. З огляду на те, що часто ЛЕП побудовані в важкодоступних місцях, послати бригаду дронів на пошук несправності десь в тайзі або в горах набагато ефективніше, ніж послати бригаду людей. Варто також відмітити використання дронів у військовій справі.

Для визначення позицій противника та кількості бойових одиниць – дрони є відмінними помічниками.

Дрони eSmart знаходять елементи енергетичної інфраструктури і в разі виявлення пошкоджень позначають об'єкт, залишаючи попередження для оператора

1.4. Методи виявлення ознак

Метод виявлення може бути реалізований за допомогою повного перебору ознак, тобто перевірити всі з можливих наборів ознак і вибрати ті ознаки, на яких похибка мінімальна. Такий метод простий у реалізації, але він абсолютно неефективний на великих даних, тому в цьому випадку найчастіше використовуються інші алгоритми. Існують три основні класи алгоритмів вибору компонентів – фільтри, обгортки і вбудовані алгоритми [4].

- *Фільтри (filters)* засновані на деяких показниках, які не залежить від методу класифікації. Наприклад такі як, кореляція ознак з цільовим вектором, критерії інформативності. Вони застосовуються до класифікації. Однією із переваг фільтрації є те, що вона може бути використана в якості попередньої обробки для зменшення розмірності простору і подолання

перенавчання. Методи фільтрації, як правило, швидко працюють. Фільтри використовуються для відбору ознак в кластеризації, для побудови початкового наближення [5]. На жаль такі методи не призначені для виявлення складних зв'язків між ознаками, і, як правило, не є достатньо чутливими для виявлення всіх залежностей в даних.

- *Вбудовані алгоритми (embedded algorithms)* виконують відбір компонент під час процедури навчання класифікатора, і саме вони явно оптимізують набір використовуваних ознак для досягнення кращої точності [6]. Переваги вбудованих алгоритмів в тому, що як правило саме вони знаходять рішення швидше, уникаючи перепідготовки даних з нуля, при цьому зникає необхідність розділяти дані на навчальну і тестову підвибірки. Разом з тим науці не відомі які-небудь вбудовані методи, що дозволяють вирішити всі існуючі завдання.
- *Методи обгортки (wrappers)* спираються на інформацію про важливість ознак, отриману від деяких методів класифікації або регресії, і тому можуть знаходити більш глибокі закономірності в даних, ніж фільтри. Обгортки можуть використовувати будь-який класифікатор, який визначає ступінь важливості ознак. Детальніше кілька алгоритмів обгортки будуть розглянуті в цій роботі далі.

1.5. Алгоритми виявлення ознак за допомогою Random Forest

Random forest (RF) – алгоритм машинного навчання, запропонований Лео Брейманом і Адель Катлер [7]. Являє собою ансамбль численних незміщених, але чутливих до навчальної вибірки алгоритмів (дерев рішень). Кожен із цих класифікаторів будується на випадковій підмножині об'єктів і випадковій підмножині ознак. Алгоритм поєднує в собі дві основні ідеї: метод беггінга Бреймана і метод випадкових підпросторів, запропонований Tin Kam Ho.

Нехай навчальна вибірка складається з N прикладів, розмірність простору ознак дорівнює M, і заданий параметр m. Запишемо покроково алгоритм Random Forest.

Всі дерева ансамблю будуються незалежно один від одного за такою процедурою:

1. Згенеруємо випадкову підвибірку (з повторенням) розміром n з навчальної вибірки.
2. Побудуємо дерево рішень, що класифікує приклади такої підвибірки. В ході створення чергового вузла дерева будемо вибирати ту ознаку, на основі якої проводиться розбиття. Конкретну ознаку обираємо не з усіх M ознак, а лише з m випадково обраних.
3. Дерево будується до повного вичерпання підвибірки і не піддається процедурі прунінгу (англ. Pruning – відсікання гілок).

Класифікація об'єктів проводиться шляхом голосування: кожне дерево ансамблю відносить об'єкт, який класифікується, до одного з класів, і перемагає клас, за який проголосувала найбільша кількість дерев.

Алгоритм Random Forest може бути використаний в завданні оцінки важливості ознак. Крім того, Random Forest має ще деякі переваги для використання його в якості алгоритму виявлення ознак: він має дуже мало параметрів, що настроюються, відносно швидко і ефективно працює, що дозволяє знаходити інформативність ознак без значних обчислювальних витрат.

1.6. Boruta

Евристичний алгоритм виявлення значущих ознак, заснований на використанні Random Forest [8]. Суть алгоритму полягає в тому, що копіюються ознаки, а потім кожна нова ознака заповнюється випадковим

чином, шляхом перетасовки його значень. На отриманій вибірці запускається Random Forest.

З метою отримання статистично значущих результатів ця процедура повторюється кілька разів, змінні генеруються незалежно на кожній ітерації.

Запишемо покроково алгоритм Boruta:

1. Додати в дані копії всіх ознак. Надалі копії будемо називати прихованими ознаками.
2. Випадковим чином перемішати приховані ознаки.
3. Запустити Random Forest і отримати Z-міру всіх ознак. Z – міра це така міра, яка обчислюється як середня втрата поділена на стандартне відхилення.
4. Знайти максимальну Z-міру з усіх Z-мір для прихованих ознак.
5. Видалити ознаки, у яких Z-міра є меншою ніж міра, знайдена на попередньому кроці.
6. Видалити всі приховані ознаки.
7. Повторювати всі кроки до тих пір поки Z-міра всіх ознак не стане більшою за максимальну Z-міру прихованих ознак.

1.7. ACE

ACE (Artificial Contrasts with Ensembles) [9] – ще один алгоритм, який може бути використаний для виявлення ознак. Головна ідея алгоритму ACE схожа з ідеєю алгоритму Boruta – кожна ознака заповнюється випадковим чином, шляхом перетасовки його значень. На отриманій вибірці запускається Random Forest. Однак в ньому, на відміну від Boruta, будуть збережені знайдені ознаки з найменшою важливістю, які дозволяють підвищити якість вимірів важливих ознак.

Найбільш важливі ознаки, знайдені алгоритмом ACE, навпаки видаляють, що дозволяє алгоритму знаходити більш тонкі зв'язки в алгоритмах.

Розділ ІІ. Алгоритм побудови класифікаторів

2.1. Опис задач побудови класифікаторів

Нехай задано сукупність лінійних функцій $f_i(x,W^i)=\langle w^i,x\rangle+w_0^i$, де $x\in R^n$ – вектор ознак, $W^i=(w^i,w_0^i)\in R^{n+1}$ – вектор параметрів, $i=1,...,m,\ m\geq 2$. Позначимо $W=(W^1,...,W^m)$, $W\in R^L$, $L=m(n+1)$. Лінійним класифікатором називається функція $a(x,W)$ наступного вигляду:

$$a(x,W)=\arg\max_i\left\{f_i(x,W^i):\ i=1,...,m\right\},\ x\in R^n,\ W\in R^L. \qquad (1)$$

При $m=2$ наведене співвідношення може бути спрощено, лінійні класифікатори описуються однією лінійною функцією $f(x,W)=\langle w,x\rangle+w_0$, $W=(w,w_0)\in R^{n+1}$, і представляються у вигляді

$$a(x,W)=\begin{cases}1, \text{ если } f(x,W)>0,\\ 2, \text{ если } f(x,W)\leq 0.\end{cases} \qquad (2)$$

Функції $f_i(x,W^i)$ зазвичай називаються дискримінант ними функціями. Вважається заданою сукупність скінченних непересічних множин (навчальна вибірка) точок з R^n: $\Omega_i=\left\{x^t:\ t\in T_i\right\}$, $i=1,...,m$, $T=\bigcup_{i=1}^{m}T_i$.

Задача побудови (навчання) класифікатора $a(x,W)$ полягає у визначенні значень параметрів W на підставі навчальної вибірки Ω_i, $i=1,...,m$, після чого функція $a(x,W)$ використовується для віднесення довільної точки $x\in R^n$ до одного з класів $1,...,m$.

Кажуть, що класифікатор $a(x,W)$ правильно розділяє точки із Ω_i, $i=1,...,m$, якщо $a(x,W)=i$, для всіх $x\in\Omega_i$, $i=1,...,m$. Нехай $i(t)$ – номер із множини Ω_i, якому належить точка x^t, $t\in T$. При $m>2$ величина

$$g^t(W)=\min\left\{f_i(x^t,W^i)-f_j(x^t,W^j):\ j\in\{1,...,m\}\setminus i,\ i=i(t)\right\}=$$

$$=\min\left\{\left\langle w^i-w^j,x^t\right\rangle+w_0^i-w_0^j:\ j\in\{1,...,m\}\setminus i,\ i=i(t)\right\} \qquad (3)$$

називається проміжком класифікатора $a(x,W)$ в точці x^t, $t \in T$.

У випадку $m = 2$ проміжком класифікатора в точці x^t являється величина

$$g^t(W) = \begin{cases} f(x^t, W), \text{если } t \in T_1, \\ -f(x^t, W), \text{ если } t \in T_2. \end{cases} \quad (4)$$

Величина $g(W) = \min\left\{g^t(W): t \in T\right\}$ називається проміжком класифікатора $a(x,W)$ на сукупності множин Ω_i, $i = 1,..., m$. Класифікатор $a(x,W)$ правильно розділяє точки із множин Ω_i, $i = 1,..., m$, якщо $g(W) > 0$.

Множини Ω_i, $i = 1,..., m$, множини називаються роздільними в класі лінійних класифікаторів, якщо існує лінійний класифікатор, який правильно розділяє точки з цих множин.

Класифікатор $a(x,W)$ інваріантний щодо множення всіх функцій f_i (векторів W^i) на додатне число, проміжок $g(W)$ лінійний щодо такої операції множення. Величину $g(W)$ можна використовувати як критерій якості класифікатора $a(x,W)$ (чим більше значення $g(W)$, тим надійніше поділяються точки з Ω_i, $i = 1,..., m$), однак, при цьому повинно враховуватися деяке нормування сукупності векторів W, яку позначимо $\eta(W)$ і будемо називати нормою класифікатора $a(x,W)$.

Будемо розглядати задачу побудови оптимального класифікатора (визначення значень параметрів W) для множин, які розділяються в класі лінійних класифікаторів, що має вигляд:

$$g^* = \max_W \left\{g(W): \eta(W) \le 1, W \in R^L\right\}. \quad (5)$$

У випадку $m > 2$ може використовуватися норма $\eta(W) = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(w_j^i)^2}$, для $m = 2$ – $\eta(W) = \sqrt{\sum_{j=1}^{n}(w_j)^2}$.

Задача (5) може бути записана в еквівалентних формах

$$\eta^* = \min_V \left\{ \eta(V) : g(V) \geq 1, V \in R^L \right\}, \quad (6)$$

$$\eta^* = \min_V \left\{ \eta(V) : g^t(V) \geq 1, t \in T, V \in R^L \right\}. \quad (7)$$

Тут еквівалентність розуміється у наступному сенсі – якщо W^* оптимальний розв'язок задачі (5), то для оптимального розв'язку V^* задачі (6) або (7) має місце [10] $V^* = W^* / g^*$, $\eta^* = 1/g^*$. Зауважимо, що $g^* > 0$ для множин, роздільних в класі лінійних класифікаторів.

2.2. Послідовні лінійні класифікатори

Для того, щоб дві скінченні множини були роздільними в класі лінійних класифікаторів, необхідно і достатньо, щоб опуклі оболонки цих множин не перетиналися. У випадку багатьох множин цих умов недостатньо. На малюнку наведено контрприклад.

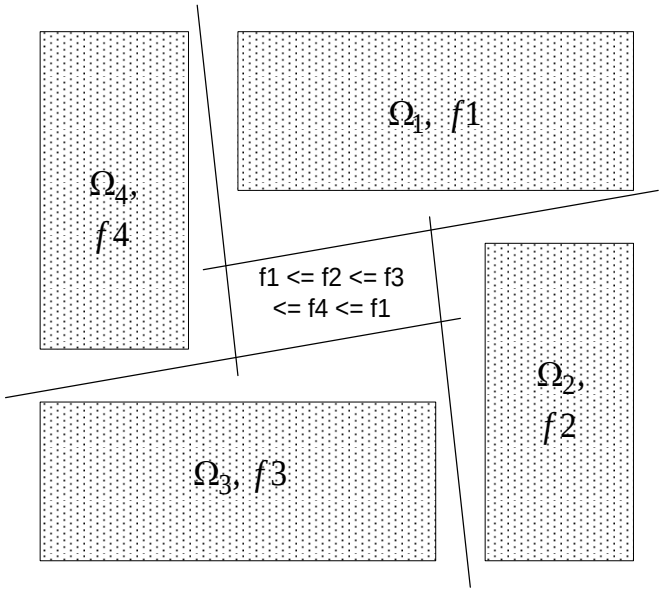


Рис. Приклад множин, нероздільних в класі лінійних класифікаторів

В [10] формулюються деякі достатні умови роздільності в класі лінійних класифікаторів для довільного числа множин. Більш поглиблений геометричний аналіз таких умов наведено в [11].

Поряд з лінійним класифікатором (1) розглянемо інший підхід, який будемо називати послідовним лінійним (бінарним) класифікатором. Нехай задано впорядкованих S пар (i, j), $i, j = 1,..., m, i \neq j,$ і для кожної пари множин Ω_i, Ω_j визначено класифікатор $a_{ij}(x, W^{ij})$ виду (2), що розділяє ці множини. Послідовний лінійний алгоритм класифікації (класифікатор) полягає в послідовному (відповідно до порядку S) застосуванні класифікаторів $a_{ij}(x, W^{ij})$ для аналізу точки x. Якщо в результаті застосування $a_{ij}(x, W^{ij})$ виявиться, що точка x належить півпростору, який містить множину Ω_i, то з подальшого аналізу альтернативна множина Ω_j і пов'язані з нею класифікатори $a_{ij}(x, W^{ij})$ повинні бути виключені. Таким чином, для класифікації точки x необхідно $m-1$ раз застосувати класифікатор виду (2) для різних пар множин $\Omega_i, \Omega_j, i, j = 1,..., m, i \neq j$.

Неважко бачити, що при довільному впорядкуванні S для роздільності множин $\Omega_i, i = 1,..., m$ в класі послідовних лінійних класифікаторів необхідно і достатньо, щоб опуклі оболонки цих множин не перетиналися. З огляду на вище сказане, отримуємо, що можливості послідовних лінійних (бінарних) класифікаторів є ширшими у порівнянні з лінійними класифікаторами виду (1).

Необхідно відзначити, що розглянуті послідовні класифікатори близькі по конструкції до DAGSVM-алгоритмів, введеними в [12].

2.3. Мінімізація емпіричного ризику

У разі лінійно нероздільної вибірки природним критерієм вибору класифікатора є мінімізація емпіричного ризику, тобто числа точок навчальної вибірки, які класифікатор розділяє неправильно.

Будемо вважати, що задано деякий параметр $\delta > 0$ надійності поділу точок навчальної вибірки Ω_i, $i = 1,..., m$. Точки x^t, $t \in T$, для яких величина зазору $g^t(W) < \delta$, поділяються класифікатором $a(x,W)$ ненадійно. Емпіричний ризик з урахуванням надійності, який визначається параметром δ, дорівнює числу точок навчальної вибірки, які класифікатор розділяє неправильно або ненадійно.

Обмежимося випадком двох класів $m = 2$. Вже згадана задача полягає у визначенні мінімальної кількості точок, які потрібно виключити з навчальної вибірки, щоб точки, які залишилися, поділялися надійно. Природно вимагати, щоб після виключення в кожному класі залишалася хоча б одна точка. Це можливо, якщо

$$\delta < \max\left\{\left\|x^\tau - x^s\right\| : \tau \in T_1,\ s \in T_2\right\}. \tag{8}$$

Надалі будемо припускати виконання цієї умови. Можна показати, що існують досить великі додатні числа B_t, $t \in T$ (в [13] передбачалося, що всі B_t однакові), при яких задача мінімізації емпіричного ризику з урахуванням надійності подана в вигляді:

$$Q^* = \min_{w,y}\left\{\sum_{t\in T} y_t\right\}, \tag{9}$$

при обмеженнях

$$g^t(W) \ge \delta - B_t \cdot y_t,\ t \in T, \tag{10}$$

$$\langle w, w\rangle \le 1, \tag{11}$$

$$\sum_{t\in T_i} y_t \le |T_i| - 1,\ \ i = 1,2, \tag{12}$$

$$0 \le y_t \le 1,\ t \in T, \tag{13}$$

$$y_t = 0 \vee 1,\ t \in T. \tag{14}$$

Змінна y_t визначає, чи враховується точка x^t при формулюванні задачі. Якщо $y_t = 1$, то точка x^t виключається з навчальної вибірки. Обмеження (12)

визначають умову того, що, по крайній мірі, одна точка з кожної множини повинна бути включена в задачу.

Задача (9)-(14) – NP-повна. У зв'язку з цим для практичного використання повинні розроблятися наближені алгоритми розв’язання такої задачі. При невеликих значеннях розмірності задачі можуть застосовуватися існуючі програмні засоби оптимізації загального призначення.

У якості наближених можуть розглядатися алгоритми, засновані на ідеях спрямованого перебору (послідовного аналізу варіантів, методу гілок і меж), локального пошуку. При розробці таких алгоритмів важливим є наявність ефективних процедур обчислення оцінок знизу величини Q^* і побудови допустимих розв’язків задачі (9) - (14).

Для реалізації цих процедур будемо використовувати неперервну релаксацію задачі (9) - (14). Зрозуміло, що всі цілочисельні формулювання задачі (9) - (14) при досить великих значеннях величин B_t еквівалентні. Однак неперервна релаксація цієї задачі і значення оцінки знизу для Q^* істотно залежать від значень B_t. Для отримання найкращої оцінки для Q^* необхідно використовувати найменші можливі значення для B_t, які позначимо $B_t^*, t \in T$.

Нехай $t \in T$, $s \in T_1$, $\tau \in T_2, s, \tau \neq t$. Розглянемо задачу

$$\beta_t^{s\tau} = \max\left\{\delta - g^t(W)\right\}, \tag{15}$$

$$g^j(W) \geq \delta,\ j = s, \tau, \tag{16}$$

$$\langle w, w \rangle \leq 1. \tag{17}$$

Лемма 1. *Справедливою є наступна рівність*

$$B_t^* = \max\left\{\beta_t^{s\tau} : s \in T_1, \tau \in T_2,\ s, \tau \neq t\right\},\ t \in T. \tag{18}$$

Доведення. Позначимо $y = (y_t, t \in T)$, Y – множина всіх y, які задовольняють обмеження (12), (14), $D(y)$ – множина всіх векторів W, які

задовольняють обмеження (10), (11) при заданому значенні вектора y. Нехай вектор $y \in Y$ такий, що $y_t = 1$. Нехай

$$\beta_t(y) = \min\left\{\theta : g^t(W) \geq \delta - \theta,\ W \in D(y)\right\} = \max\left\{\delta - g^t(W) : W \in D(y)\right\}.$$

Очевидно, що $B_t^* = \max\{\beta_t(y) : y \in Y,\ y_t = 1\}$. Нехай $s \in T_1$, $\tau \in T_2$, $s, \tau \neq t$. Позначимо $y^{s\tau} = \{y_t,\ t \in T,\ y_s = 0,\ y_\tau = 0,\ y_j = 1,\ j \neq s, \tau\}$. Легко побачити, що для будь-якого $y \in Y$ такого, що $y_s = 0,\ y_\tau = 0$, справедливо $D(y) \subseteq D(y^{s\tau})$, тобто $\beta_t(y) \leq \beta_t(y^{s\tau})$. Звідки $B_t^* = \max\left\{\beta_t(y^{s\tau}) : s \in T_1, \tau \in T_2,\ s, \tau \neq t\right\}$, і враховуючи, що $\beta_t^{s\tau} = \beta_t(y^{s\tau})$ – отримуємо твердження леми. ■

Нехай $t \in T_1$. Розглянемо більш детально задачу (15)–(17). Враховуючи (4), перепишемо цю задачу у вигляді

$$\beta_t^{s\tau} = -\min_{w, w_0}\left\{\left\langle w, x^t\right\rangle + w_0 - \delta\right\}, \tag{19}$$

$$\left\langle w, x^s\right\rangle + w_0 \geq \delta,\ s \in T_1, \tag{20}$$

$$-\left\langle w, x^\tau\right\rangle - w_0 \geq \delta,\ \tau \in T_2, \tag{21}$$

$$\langle w, w\rangle \leq 1. \tag{22}$$

Якщо система обмежень (20) – (22) несумісна, то $\beta_t^{s\tau} = -\infty$. Це справедливо, якщо $\delta > \left\|x^s - x^\tau\right\|$. В силу (8) завжди існує пара s, τ, така, що $\delta \leq \left\|x^s - x^\tau\right\|$.

Легко бачити, що в оптимальному розв’язку задачі (19) - (22) обмеження (20), (22) обов'язково виконуються як рівності, а обмеження (21) може бути як активним, так і неактивним. Розглянемо випадок, коли обмеження (21) неактивне в оптимальному розв’язку. Використовуючи правило множників Лагранжа, для оптимального розв’язку отримуємо

$$w = \frac{x^s - x^t}{\left\|x^s - x^t\right\|},\quad w_0 = \delta - \left\langle w, x^s\right\rangle,\quad \beta_t^{s\tau} = \left\|x^s - x^t\right\|.$$

При цьому для отриманого вектора (w, w_0) має виконуватися обмеження (21). Якщо це обмеження не виконується, то оптимальний розв'язок має будуватися з урахуванням того, що обмеження (21) активне.

Отримані співвідношення дозволяють порівняно просто визначати значення $B_t^*, t \in T$.

Розглянемо задачу (9)-(13) – неперервну релаксацію задачі мінімізації емпіричного ризику. Нехай $d^t(W) = \max\left(0, \frac{1}{B_t}\left(\delta - g^t(W)\right)\right)$. Зафіксуємо деякі значення змінних W. Легко побачити, що якщо при цих значеннях W існує розв'язок задачі (9)-(13), то $y^t = d^t(W)$. Звідки і отримуємо задачу мінімізації за змінними W:

$$q^* = \min_W \sum_{t \in T} d^t(W) \tag{23}$$

при обмеженнях

$$\langle w, w \rangle \le 1, \tag{24}$$

$$\sum_{t \in T_i} d^t(W) \le |T_i| - 1,\ i = 1, \ldots, m, \tag{25}$$

$$d^t(W) \le 1,\ t \in T. \tag{26}$$

Величина q^* є оцінкою знизу для мінімального значення емпіричного ризику Q^*, а вектор W, отриманий в результаті розв'язку задачі (23)–(26), визначає наближений розв'язок задачі (9)–(14). Функції $d^t(W)$ – випуклі кус очно-лінійні. Для розв'язку задачі (23)-(26) доцільно застосувати ефективні методи негладкої оптимізації [14].

2.4. Метод опорних векторів

Метод опорних векторів (англ. SVM, support vector machine) – набір схожих алгоритмів навчання з учителем, що використовуються для задач

класифікації та регресійного аналізу. Метод SVM належить сімейству лінійних класифікаторів і може також розглядатися як спеціальний випадок регуляризації по Тихонову. Особливою властивістю методу опорних векторів є невпинне зменшення емпіричної помилки класифікації і збільшення зазору. Тому метод також відомий як метод класифікатора з максимальним зазором.

Основна ідея методу – переведення вихідних векторів в простір більш високої розмірності і пошук розділяючої гіперплощини з максимальним зазором в цьому просторі. Дві паралельні гіперплощини будуються по обидва боки гіперплощини, що розділяє класи. Розділяючою гіперплощиною буде гіперплощина, яка максимізує відстань до двох паралельних гіперплощин. Алгоритм працює в припущенні, що чим більша різниця або відстань між цими паралельними гіперплощинами, тим меншою буде середня помилка класифікатора.

У методі опорних векторів (SVM) для випадку $m = 2$ розв'язується задача, яка може бути представлена у наступному вигляді:

$$\eta^* = \min_{v, v_0} \left\{ \langle v, v \rangle + C \sum_{t \in T} \xi^t \right\}, \tag{27}$$

$$\langle v, x^t \rangle + v_0 \geq 1 - \xi^t, \ t \in T_1, \tag{28}$$

$$-\langle v, x^t \rangle - v_0 \geq 1 - \xi^t, \ t \in T_2, \tag{29}$$

$$\xi^t \geq 0, t \in T. \tag{30}$$

Метод опорних векторів (SVM) використовується для побудови оптимального класифікатора як для лінійно роздільних класів, так і для лінійно нероздільних класів.

У випадку лінійно роздільних класів з теорем про негладкі штрафи випливає, що при досить великому коефіцієнті C задачі (7) і (27) - (30) мають однакові розв'язки. У випадку лінійно нероздільних класів задачі (27) - (30) інтерпретується [15] як деяка регуляризація задачі мінімізації емпіричного ризику.

Покажемо, що між задачею (27) - (30) і неперервною релаксацією (9) - (13) задачею мінімізації емпіричного ризику існують певні взаємозв'язки.

Ослабимо обмеження (10), поклавши $B_t = B := \max\limits_{\tau} B_\tau^*$, та виключимо обмеження (12). Задача набуде вигляду

$$\bar{q}^* = \min_{w,y}\left\{\sum_{t\in T} y_t\right\}, \tag{31}$$

при обмеженнях

$$\left\langle w, x^t\right\rangle + w_0 \geq \delta - B\cdot y_t,\ t\in T_1, \tag{32}$$

$$-\left\langle w, x^t\right\rangle - w_0 \geq \delta - B\cdot y_t,\ t\in T_2, \tag{33}$$

$$\langle w, w\rangle \leq 1, \tag{34}$$

$$y_t > 0,\ t\in T. \tag{35}$$

Тут обмеження (11) замінено парою еквівалентних обмежень (32), (33). Зрозуміло, що $\bar{q}^* \leq q^*$. Зробимо заміну змінних $w = \delta v,\ w_0 = \delta v_0$, $\xi^t = \dfrac{By_t}{\delta}$, $t\in T_1 \bigcup T_2$. Задача приймає вигляд

$$\bar{q}^* = \frac{\delta}{B}\cdot \min_{v,v_0,\xi}\left\{\sum_{t\in T}\xi^t\right\} \tag{36}$$

при обмеженнях

$$\left\langle v, x^t\right\rangle + v_0 \geq 1-\xi^t,\ t\in T_1, \tag{37}$$

$$-\left\langle v, x^t\right\rangle - v_0 \geq 1-\xi^t,\ t\in T_2, \tag{38}$$

$$\langle v, v\rangle \leq {}^{1}\!\!\diagup\!\!{}_{\delta^2}, \tag{39}$$

$$\xi_t > 0,\ t\in T. \tag{40}$$

Нехай $\alpha \geq 0$ – двоїста змінна для обмеження (39). Розглянемо функцію Лагранжа $L(\alpha,\xi,v) = \dfrac{\delta}{B}\sum\limits_{t\in T}\xi^t + \alpha\cdot(\langle v, v\rangle - {}^{1}\!\!\diagup\!\!{}_{\delta^2})$ і Лагранжеву релаксацію задачі (36)–(40):

$$\varphi(\alpha) = \min_{v,v_0,\xi} L(\alpha,\xi,u) \tag{41}$$

при обмеженнях (37), (38), (40).

Оскільки $\varphi(\alpha)$ – оптимальне значення Лагранжевої релаксації задачі (36)–(40), то $\varphi(\alpha) \le \bar{q}^*$ для будь-якого $\alpha \ge 0$. Нехай задано штрафний коефіцієнт C в задачі (27)-(30). Вибираючи α із умови $\frac{\delta}{\alpha B} = C$, отримуємо

$$L(\alpha, \xi, u) = \alpha \left\{ \langle v, v \rangle + C \cdot \sum_{t \in T} \xi^t \right\} - \frac{\alpha}{\delta^2},$$

тобто задача (41), (37), (38), (40) еквівалентна задачі (27)-(30) з точністю до адитивної константи та фіксованого множника в цільовій функції при вказаному виборі значення двоїстої змінної.

Таким чином, задача (27)-(30), яка розв’язується в методі опорних векторів, може бути отримана в результаті в результаті послаблення обмежень задачі (23)–(26), яка в свою чергу є неперервною релаксацією задачі мінімізації емпіричного ризику.

Розділ ІІІ. Метод виявлення ознак HOG

3.1. Опис методу

Гістограма направлених градієнтів (англ. Histogram of Oriented Gradients, HOG) – дескриптори особливих точок, які використовуються в комп'ютерному баченні і обробці зображень з метою розпізнавання об'єктів. Дана техніка заснована на підрахунку кількості напрямків градієнта в локальних областях зображення. Цей метод схожий на гістограми напрямлення краю, дескриптори SIFT і контексти форми, але відрізняється тим, що обчислюється на щільній сітці рівномірно розподілених осередків і використовує нормалізацію перекриваючого локального контрасту для збільшення точності.

Навніт Далал і Білл Тріггс, дослідники INRIA, вперше описали гістограму направлених градієнтів в своїй роботі на CVPR в червні 2005 року. У роботі [16] вони використовували алгоритм для знаходження пішоходів на статичних зображеннях, хоча згодом розширили область застосування до знаходження людей на відео, а також різних тварин і машин на статичних зображеннях.

Основною ідеєю алгоритму є припущення, що зовнішній вигляд і форма об'єкта на ділянці зображення можуть бути описані розподілом градієнтів інтенсивності або направленням країв. Реалізація цих дескрипторів може бути проведена шляхом поділу зображення на маленькі зв'язкові області, іменовані осередками, і розрахунком для кожного осередку гістограми направлень градієнтів або напрямків країв для пікселів, що знаходяться всередині осередку. Комбінація цих гістограм і є дескриптором. Для збільшення точності локальні гістограми піддаються нормалізації по контрасту. З цією метою обчислюється міра інтенсивності на великому фрагменті зображення, який називається блоком, і отримане значення

використовується для нормалізації. Нормалізовані дескриптори мають кращу інваріантність по відношенню до висвітлення.

Дескриптор HOG має кілька переваг над іншими дескрипторами. Оскільки HOG працює локально, метод підтримує інваріантність геометричних і фотометричних перетворень, за винятком орієнтації об'єкта. Подібні зміни з'являться тільки в великих фрагментах зображення. Більш того, як виявили Далал і Тріггс, грубе розбиття простору, точне обчислення напрямків і сильна локальна фотометрична нормалізація дозволяють ігнорувати рух пішоходів, якщо вони підтримують вертикальне положення тіла. Дескриптор HOG, таким чином, є хорошим засобом знаходження людей на зображеннях [16].

3.2. Реалізація алгоритму

3.2.1. Обчислення градієнта

Першим кроком обчислень у багатьох детекторах особливих точок є нормалізація кольору і гамма-корекція. Далал і Тріггс встановили, що для дескриптора HOG цей крок можна опустити, оскільки подальша нормалізація дасть той же результат. Тому на першому етапі розраховуються значення градієнтів. Найпоширенішим методом є застосування одновимірної диференціюючої маски в горизонтальному і / або вертикальному напрямку. Цей метод вимагає фільтрації (колірної або складової яскравості) за допомогою наступних фільтруючих ядер:

$$[-1,0,1] \text{ та } [-1,0,1]^{\mathrm{T}}.$$

Далал і Тріггс використовували більш складні маски, такі як Собел 3x3 (Оператор Собеля) або діагональні маски, але ці маски показали нижчу продуктивність для даного завдання. Вони також експериментували з розмиванням по Гаусу перед застосуванням диференціюючої маски, але також виявили, що пропуск цього кроку збільшує швидкодію без помітної втрати якості [17].

3.2.2. Групування напрямків

На наступному кроці обчислюються гістограми осередків. Кожен піксель в осередку бере участь у зваженому голосуванні для каналів гістограми напрямків, який ґрунтується на значенні градієнтів. Осередки можуть бути прямокутної або круглої форми, канали гістограми рівномірно розподіляються від 0 до 180 або ж від 0 до 360 градусів, в залежності від того, обчислюється «знаковий» або «беззнаковий градієнт». Далал і Тріггс виявили, що беззнаковий градієнт спільно з дев'ятьма каналами гістограми дає кращі результати при розпізнаванні людей. При розподілі ваг в голосуванні вага пікселя може задаватися або абсолютним значенням градієнта, або деякою функцією від нього; в реальних тестах абсолютне значення градієнта дає кращі результати. Іншими можливими варіантами можуть бути квадратний корінь, квадрат або урізане абсолютне значення градієнта.

3.2.3. Блоки дескрипторів

Для прийняття до уваги яскравості і контрастності градієнти слід локально нормувати, для чого осередки потрібно згрупувати в більш великі зв'язкові блоки. Дескриптор HOG, таким чином, є вектором компонент нормованих гістограм осередків з усіх областей блоку. Як правило, блоки перекриваються, тобто кожен осередок входить більш ніж в один кінцевий дескриптор. Використовуються дві основні геометрії блоку: прямокутні R-HOG і круглі C-HOG. Блоки R-HOG зазвичай є квадратними сітками, що характеризуються трьома параметрами: кількістю осередків на блок, кількістю пікселів на осередок і кількістю каналів на гістограму осередка. В експерименті дала і Тріггса оптимальними параметрами є блоки 16x16, осередки 8x8 і 9 каналів на гістограму. Більш того, вони виявили, що можна злегка підвищити швидкість обчислень, застосовуючи гаусів фільтр всередині кожного блоку до процедури голосування, що, в свою чергу, знижує вагу пікселів на границях блоків. Блоки R-HOG виявляються дуже схожими на SIFT-дескриптори; однак, незважаючи на їх схожу структуру,

блоки R-HOG обчислюються на щільних сітках фіксованого масштабу без фіксованого напрямку, в той час як SIFT-дескриптори обчислюються в розріджених, що не є чутливими до масштабу ключових точках зображення і використовують поворот для вирівнювання напрямку. Крім того, для кодування інформації про форму об'єктів блоки R-HOG використовуються спільно, в той час як SIFT-дескриптори використовуються окремо.

Блоки C-HOG мають 2 різновиди: з цілим центральним осередком і розділеною на сектори. Ці блоки можуть бути описані 4 параметрами: кількість секторів і кілець, радіус центрального кільця і коефіцієнт розширення для радіусів інших кілець. Дала і Тріггс виявили, що обидва різновиди показали однаковий результат, і поділ на 2 кільця і 4 сектори з радіусом 4 пікселя і коефіцієнтом розширення 2 дало кращий результат в їхньому експерименті. Крім того, гаусове зважування не дало ніяких покращень при використанні блоків C-HOG. Ці блоки схожі на контексти форми, але мають важливу відмінність: блоки C-HOG містять осередки з декількома каналами напрямків, в той час як контексти форми використовують тільки наявність одного краю [17].

3.3. Нормалізація блоків

Далал і Тріггс дослідили чотири методи нормалізації блоків. Нехай v – ненормований вектор, який містить всі гістограми даного блоку. $\|v\|_k$ – його k-норма при $k = 1,2$ і e – деяка мала константа (точне значення є не важливим). Тоді нормований множник можна отримати одним із наступних способів:

$$L_2 - \text{норма:} \qquad f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}.$$

$L_2 -$ норма обмежується зверху (значення v більші за 0,2 вважаються рівними 0,2) і перенормовується.

$$L_1 - \text{норма:} \qquad f = \frac{v}{(\|v\|_1 + e)}.$$

Далал і Тріггс встановили, що L_1-норма дає менш надійні результати, ніж інші. Кінцевим кроком в розпізнаванні об'єктів з використанням HOG є класифікація дескрипторів за допомогою системи навчання з учителем. Далал і Тріггс використовували метод опорних векторів (SVM, Support Vector Machine).

Розділ IV. Програмна реалізація алгоритму виявлення обличчя на зображенні

4.1. Опис вхідних даних

Бібліотека Scickit-Learn містить базу даних Wild, яка у свою чергу містить 13233 зображень людських облич. Ці зображення будуть використані у якості позитивної навчальної вибірки. Окрім позитивних прикладів необхідні також приклади зображень предметів (без зображення людських облич на них), які виступатимуть негативною навчальною вибіркою. Для цього використаємо зображення, що постачаються разом із Scikit-Image у Scikit-Learn, з допомогою PatchExtractor.

Для оцінки якості роботи алгоритму виявлення ознак HOG та класифікатора отримаємо точність роботи алгоритму, а також встановимо найкращий класифікатор. Для перевірки роботи програми (визначення обличчя на зображенні) використаємо довільні зображення, отримані з інтернету.

4.2. Опис програмного середовища

Згідно з індексом TIOBE (щомісячний індикатор популярності мов програмування на базі підрахунків результатів пошукових запитів) Python три рази визначався мовою року: в 2007, 2010 і 2018. Нагорода присуджується мові програмування, яка має найвищий ріст рейтингу за рік [18].

Цікаво, що в березні цього року Python зайняв свою найвищу позицію в рейтингу з 2001 року. Згідно TIOBE Index зараз він знаходиться на 3 місці.

Як і будь-яка інша мова програмування, Python має плюси і мінуси. Однак кількість розробників, захоплених цією мовою програмування, зростає так само, як і кількість проектів, які вимагають кваліфікованих Python-

фахівців. «Не потрібно винаходити черговий велосипед» – так говорять багато розробників про Python.

Основні переваги Python:

- **Низький поріг входження.** Синтаксис Python більш зрозумілий для новачка у порывнянны з іншими мовами програмування.
- **Логічна, лаконічна і зрозуміла.** У порівнянні з багатьма іншими мовами, Python має легкочитаємий синтаксис.
- **Крос-платформленість.** Python підходить для різних платформ: і Linux, і Windows.
- Існує реалізація інтерпретаторів для мобільних пристроїв і непопулярних систем.
- **Широке застосування**. Використовується для розробки веб-додатків, ігор, мова зручна для автоматизації, математичних обчислень, машинного навчання. Існує реалізація під назвою Micro Python, оптимізована для запуску на мікроконтролерах (можна писати інструкції, логіку взаємодії пристроїв, організовувати зв'язок, реалізовувати розумний будинок).
- **Сильне ком'юніті і багато конференцій.** Наприклад, недавно в Одесі відбувся PyCon. На конференції в числі всіх спікерів виступили 4 іноземних доповідача, які розглянули цікаві теми, серед яких – розпізнавання образів з допомогою машинного навчання засобами Python.
- **Потужна підтримка компаній-гігантів IT-індустрії.** Такі компанії, як Google, Facebook, Dropbox, Spotify, Quora, Netflix, на певних етапах розробки використовували саме Python.
- **Висока затребуваність на ринку праці.**

У світі Python багато якісних бібліотек, так що не потрібно винаходити велосипед, якщо треба терміново вирішити якесь завдання. Для навчання є багато хороших книг, в першу чергу англійською мовою, звичайно, але і в перекладі також можна знайти гідну літературу.

Python відрізняється суворою вимогою до написання коду (відступи), що є великого перевагою. Мова сама сприяє писати код організовано і красиво.

Python розвивається і не згасне ще довго. За численними оглядами і рейтингами мова Python займає високі позиції. Згідно DOU вона знаходиться на п'ятому місці і займає третю позицію у веб-технологіях.

Якщо говорити про мінуси, то Python – це мова з динамічною типізацією. З одного боку код простіше і швидше писати, але продуктивність поступається таким компільованим мовам, як C ++ і Golang.

Але для більшості завдань: веб-розробки, скриптів, аналізу даних, машинного навчання та роботи з великими даними Python – одина з кращих мов.

4.3. Scikit-learn

Бібліотека Scikit-learn – найпоширеніший інструмент для вирішення задач класичного машинного навчання. Вона представляє собою широкий вибір алгоритмів навчання з учителем і без вчителя. Навчання з учителем передбачає наявність розміченого набору даних, в якому відомі значення цільового показника. У той час як навчання без вчителя не передбачає наявності індикаторів або розмітки в наборі даних тобто, потрібно навчитися отримувати корисну інформацію з довільних вхідних даних. Одна з основних переваг бібліотеки Scikit-learn полягає в тому, що вона працює на основі декількох поширених математичних бібліотек, і легко інтегрує їх одна з одною. Ще однією перевагою є широка інтернет-спільнота користувачів та професіоналів та докладна документація. Scikit-learn широко використовується для промислових систем, в яких застосовуються алгоритми класичного машинного навчання; для досліджень; для новачків, які тільки роблять перші кроки в області машинного навчання [19].

Для своєї роботи, Scikit-learn використовує такі популярні бібліотеки:

- **NumPy**: математичні операції та операції над тензорами
- **SciPy**: науково-технічні обчислення
- **Matplotlib**: візуалізація даних
- **IPython**: інтерактивна консоль для Python
- **SymPy**: символьна математика
- **Pandas**: обробка, маніпуляції та аналіз даних

До завдань бібліотеки Scikit-learn не входить завантаження, обробка, маніпуляція з даними та їх візуалізація. З цими завданнями відмінно справляються бібліотеки Pandas і NumPy. Scikit-learn спеціалізується на алгоритмах машинного навчання для вирішення задач навчання з учителем: класифікації (прогнозування ознаки, допустимі значення якої обмежені) і регресії (прогнозування ознаки за попередніми значеннями), а також для задач навчання без учителя: кластеризації (розбиття даних по класах, які модель визначить сама), зниження розмірності (подання даних в просторі меншої розмірності з мінімальними втратами корисної інформації) і визначення аномалій в даних.

Бібліотека Scikit-learn реалізує наступні основні методи:

- **Лінійні:** моделі, завдання яких побудувати розділяючу (для класифікації) або апроксимуючу (для регресії) гіперплощину.
- **Метричні:** моделі, які обчислюють відстань по одній з метрик між об'єктами вибірки, і приймають рішення в залежності від цієї відстані (K найближчих сусідів).
- **Дерева рішень:** навчання моделей, що базуються на умовах, оптимально обраних для вирішення задачі.
- **Ансамблеві методи:** методи, засновані на деревах, що базуються на умовах, оптимально обраних для вирішення задач, які комбінують безліч дерев, і таким чином підвищують їх якість роботи, а також дозволяють проводити відбір ознак (бустінг, беггінг, випадковий ліс, мажоритарне голосування).

- **Нейронні мережі:** комплексний нелінійний метод для задач регресії і класифікації.
- **SVM:** нелінійний метод, який навчається визначати межі прийняття рішень.
- **Наївний Байес:** пряме ймовірнісне моделювання для задач класифікації.
- **PCA:** лінійний метод зниження розмірності і відбору ознак.
- ***t*-SNE:** нелінійний метод зниження розмірності.
- ***K*-середніх:** найпоширеніший метод для кластеризації, який вимагає на вхід число кластерів, на які повинні бути розподілені дані.
- **Крос-валідація:** метод, при якому для навчання використовується весь набір даних (на відміну від розбиття на вибірки train/test), проте навчання відбувається багаторазово, і в якості валідаційної вибірки на кожному кроці виступають різні частини даних. Підсумковий результат являє собою усереднення отриманих результатів.
- **Grid Search:** метод для знаходження оптимальних гіперпараметрів моделі шляхом побудови сітки зі значень гіперпараметрів і послідовного навчання моделей з усіма можливими комбінаціями гіперпараметрів з сітки.

Це – лише базовий список. Крім цього, Scikit-learn містить функції для розрахунку значень метрик, вибору моделей, попередньої обробки даних та інше.

4.4. Покрокова реалізація HOG алгоритму

Дескриптор ознак – це представлення зображення або частини зображення таким чином, щоб спростити його шляхом вилучення корисної інформації та виключення сторонніх відомостей.

Зазвичай дескриптор об’єкта перетворює зображення шириною x та висотою x 3 (канали) у вектор об’єкта/масив довжиною n. У випадку дескриптора функції HOG, вхідне зображення має розмір 64 на 128, а вихідний вектор ознак має довжину 3780.

Дескриптор HOG можна розрахувати і для інших розмірів. Вказані розміри представлені в оригінальній роботі авторів алгоритму. Яку інформацію вважатимемо «корисною», а яку «сторонньою»? Щоб визначити “корисну” інформацію, потрібно знати, для чого вона “корисна”? Очевидно, що вектор функцій не є «корисним» для перегляду зображення. Основна мета вектора функцій – розпізнавання зображень та виявлення об’єктів. Після реалізації HOG алгоритми класифікації зображень, наприклад, Support Vector Machine (SVM), дають хороші результати.

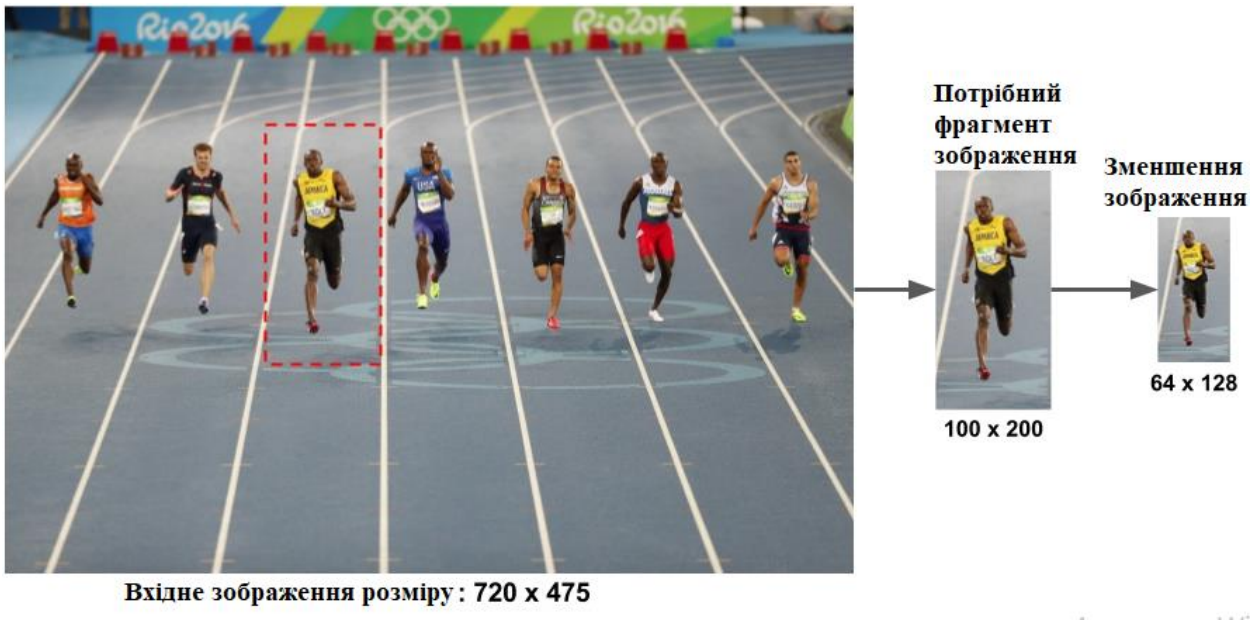
У дескрипторі OG в якості ознак використовується розподіл (гістограми) напрямків градієнтів (орієнтовані градієнти). Градієнти (похідні x та y) зображення – корисні, оскільки величина градієнтів є більшою навколо країв та кутів (області різких змін інтенсивності). Зрозуміло, що краї та кути містять набагато більше інформації про форму об’єкта, ніж плоскі області зображення.

Крок 1. Попередня обробка зображення

Як згадувалося раніше, дескриптор функції HOG обчислюється на фрагменті зображення розміром 64×128. Звичайно, зображення може бути будь-якого розміру. Єдиним обмеженням є те, що аналізовані частини зображення повинні мати мають фіксоване співвідношення сторін – 1: 2. Наприклад, вони можуть мати розміри 100×200, 128×256 або 1000×2000, але не 101×205.

Розглянемо зображення розміром 720×475. Виберемо частину зображення розміром 100×200 для розрахунку нашого дескриптора функцій HOG. Вибраний фрагмент обрізається із зображення та зменшується

до розміру 64 × 128. Тепер можна приступати до розрахунку дескриптора HOG для цього фрагменту зображення.



У роботі Далала та Трігса також згадується гамма-корекція як крок попередньої обробки, але підвищення продуктивності незначне, і тому пропускаємо цей крок.

Крок 2. Обчислення градієнтів

Для обчислення дескриптора HOG необхідно спочатку обчислити горизонтальний та вертикальний градієнти. Після цього обчислюється гістограма градієнтів. Цього легко досягти, відфільтрувавши зображення за допомогою наступних ядер.

-1	0	1

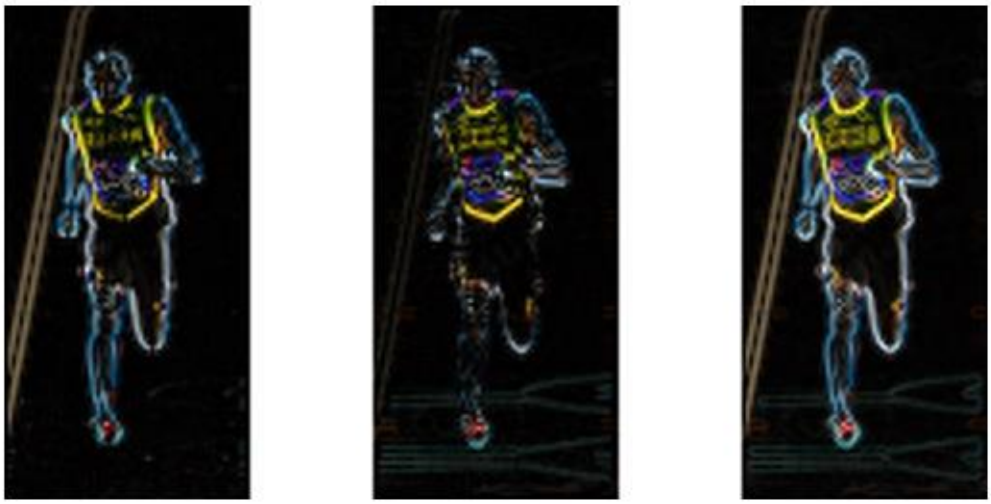
-1
0
1

Аналогічних результатів можна досягти, використовуючи оператор Собеля. Далі, знаходимо величину та напрямок градієнта, використовуючи наступні формули:

$$g = \sqrt{g_x^2 + g_y^2},$$

$$\theta = arctan\frac{g_y}{g_x}.$$

Наступне зображення показує результат роботи градієнтів.



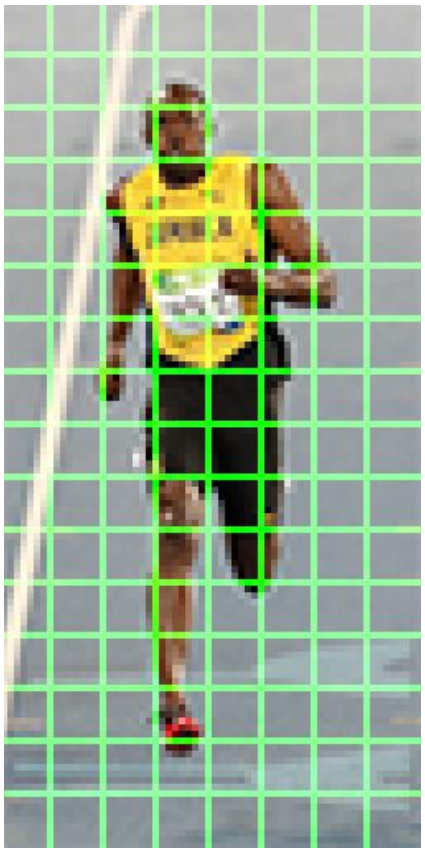
Ліворуч – абсолютне значення x-градієнта. Центр – абсолютне значення y-градієнта. Праворуч – величина градієнта.

Зауважимо, що градієнт x більш помітний на вертикальних лініях та градієнт y – на горизонтальних лініях. Величина градієнта (праворуч) поєднує роботу градієнтів x та y. Жоден із градієнтів них не відображається на гладких фрагментах зображення. Градієнтне зображення видалило багато несуттєвої інформації (наприклад, постійний кольоровий фон) та виділило контури. Іншими словами, з градієнтного зображення можна легко побачити, що на зображенні є людина.

На кожному пікселі градієнт має величину та напрямок. Для кольорових зображень оцінюються градієнти трьох каналів (як показано на малюнку вище). Величина градієнта в пікселі – це максимум величини градієнтів трьох каналів, а кут – це кут, що відповідає максимальному градієнту.

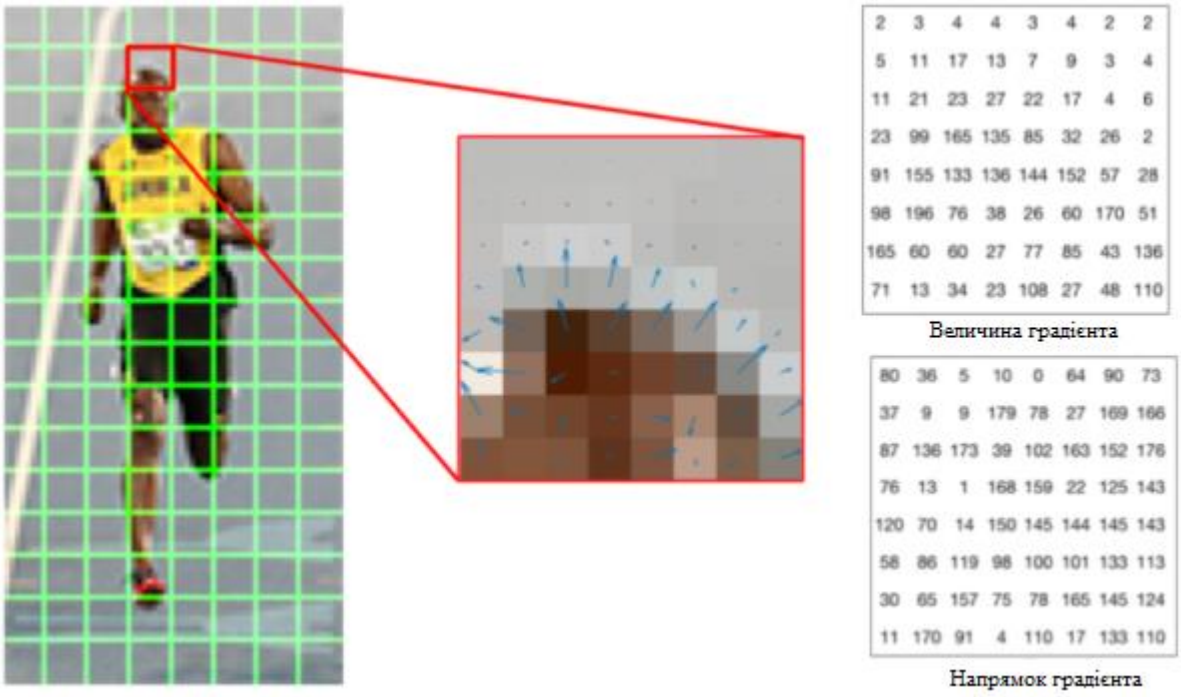
Крок 3. Обчислення гістограми градієнтів у комірках $\mathbf{8 \times 8}$

На цьому кроці зображення ділиться на $8 \times 8 = 64$ комірки і для кожної з 64 комірок розраховується гістограма градієнтів.



З’ясуємо, чому необхідно поділити зображення саме на 64 однакових клітинки. Однією з важливих причин використання дескриптора ознак для опису ділянки зображення є те, що він забезпечує компактне представлення. Частина зображення 8 × 8 містить $8 \times 8 \times 3 = 192$ пікселі. Градієнт цього фрагмента містить 2 значення (величину та напрямок) на піксель, тобто $8 \times 8 \times 2 = 128$ чисел. Далі у роботі буде показано, як ці 128 чисел представлені за допомогою гістограми 9-bin, яку можна представити як масив із 9 чисел. Окремі клітинки можуть включати шум, але гістограма розміром 8×8 робить зображення набагато менш чутливим до шуму.

Отже, розмір фрагменту 8×8 зумовлений масштабом функцій. Спочатку HOG використовувався для виявлення пішоходів. Клітинки розміру 8×8 на фотографіях пішохода в масштабі 64×128 достатньо великі, щоб захопити потрібні риси (наприклад, обличчя, маківка тощо). Гістограма – це, по суті, вектор (або масив) із 9 бінів (чисел), що відповідають кутам 0, 20, 40, 60… 160. Розглянемо один фрагмент 8×8 на зображенні і побачимо, як виглядають градієнти.



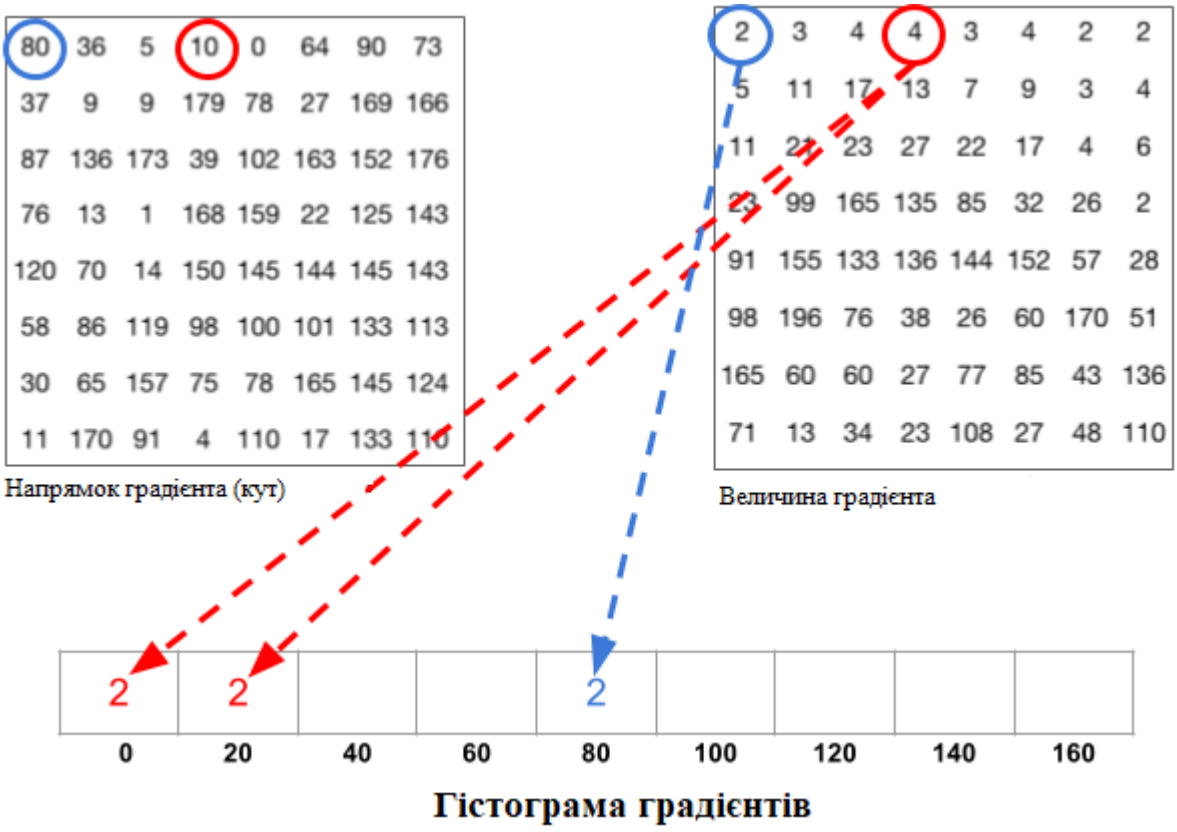
Центр: схема RGB та градієнти, представлені стрілками. Праворуч: градієнти в одній клітинці, представлені як цифри

Зображення в центрі – дуже інформативне. На ньому зображено клітинку зображення, з'єднану червоними лініями. Стрілки показують напрямок градієнта, а їх довжина – величину градієнта. Напрямок стрілок вказує на напрямок зміни інтенсивності, а товщина – наскільки велика різниця.

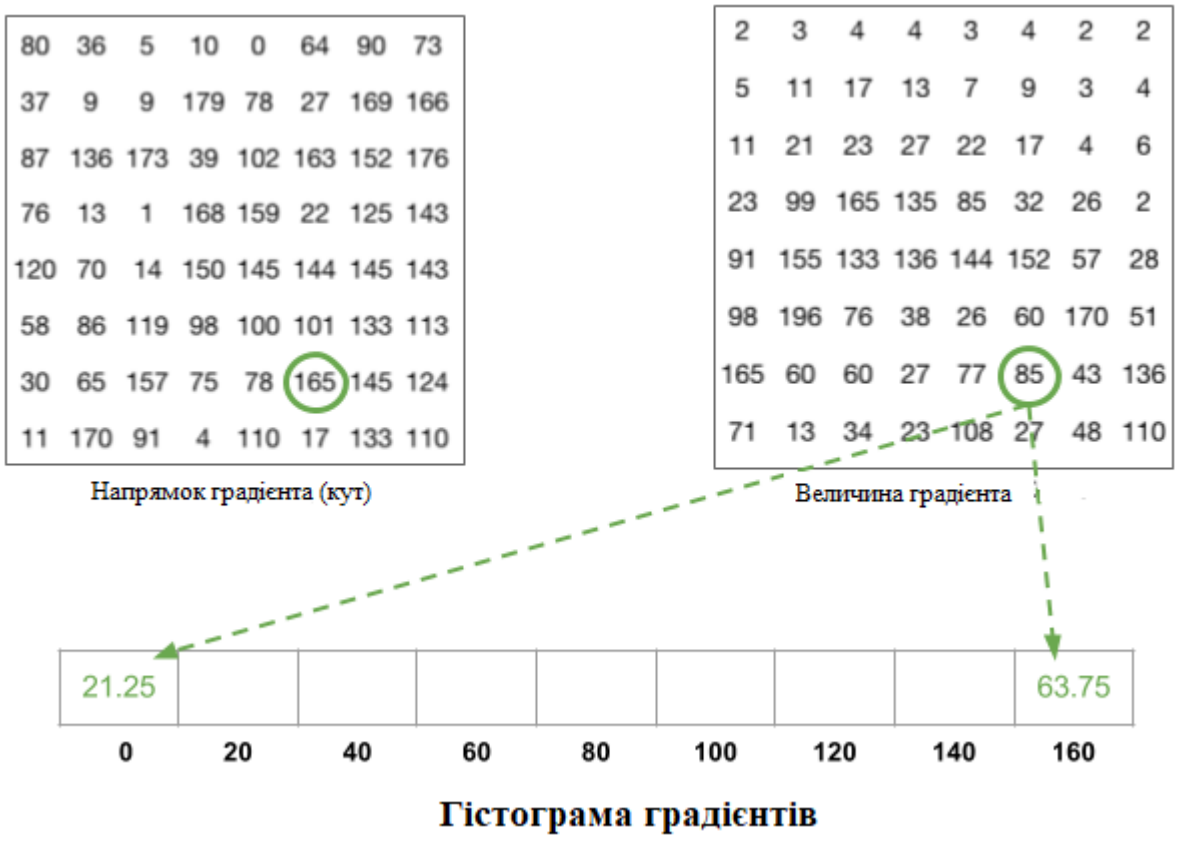
Справа попереднього рисунка знаходяться необроблені числа, що представляють градієнти в клітинках 8×8 з однією незначною різницею – кути знаходяться в межах від 0 до 180 градусів замість від 0 до 360 градусів. Вони називаються "беззнаковими" градієнтами, оскільки градієнт і йому протилежний представлені однаковими числами. Іншими словами, градієнтна стрілка та протилежна їй на 180 градусів вважаються однаковими. Справа в тому, що емпірично було показано, що непідписані градієнти працюють краще, ніж градієнти зі знаком у задачі виявлення пішоходів. Деякі реалізації HOG дозволять вказувати необхідність використання підписаних градієнтів.

Наступним кроком є створення гістограми градієнтів у цих клітинках 8 × 8. Гістограма містить 9 бункерів, що відповідають кутам 0, 20, 40 ... 160.

Наступний малюнок ілюструє процес. Розглядаємо величину та напрямок градієнта тієї самої клітинки 8 × 8, що і на попередньому малюнку. Бін вибирається на основі напряму, а значення, яке надходить у бін, вибирається на основі величини. Спершу зупинимось на пікселі, оточеному синім кольором. Він має кут (напрямок) 80 градусів і величину 2. Отже, він додає 2 до 5-ї комірки гістограми градієнтів. Градієнт на пікселі, оточеному червоним кольором, має кут 10 градусів і величину 4. Оскільки 10 градусів знаходиться на половині шляху від 0 до 20, то значення за пікселем рівномірно розподіляється на два біни.

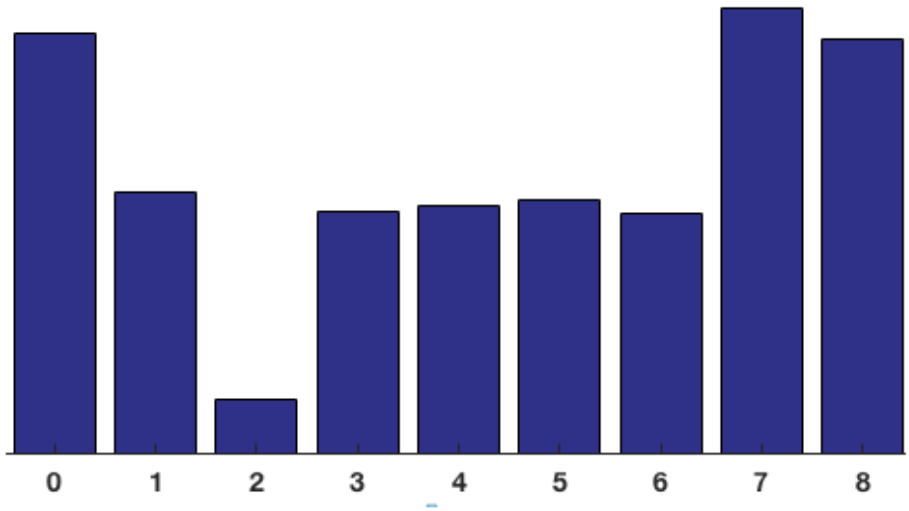


Ще одна деталь, про яку слід пам’ятати. Якщо кут перевищує 160 градусів (160 – 180 градусів), то у наведеному нижче прикладі піксель з кутом 165 градусів пропорційно вносить значення у 0 градусів бін та 160 градусів бін.



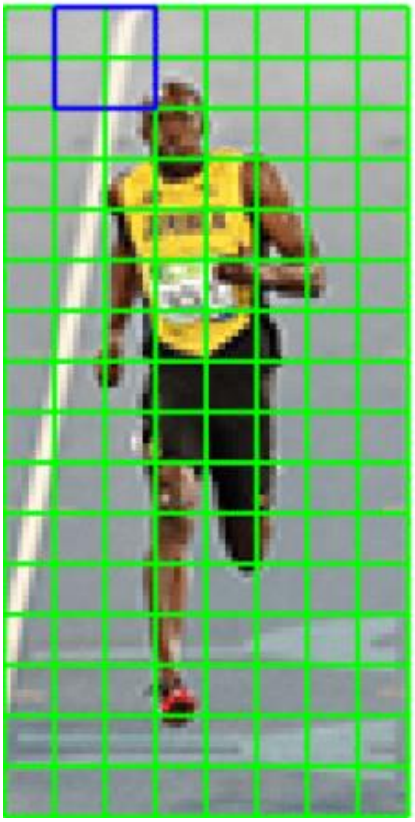
Гістограма градієнтів

Таким чином будується 9-ти бінова гістограма, яка в даному випадку виглядає наступним чином:



Вісь *y* дорівнює 0 градусам. В обраній клітинці градієнти спрямовані вгору або вниз.

Крок 4. Нормалізація блоків $\mathbf{16 \times 16}$



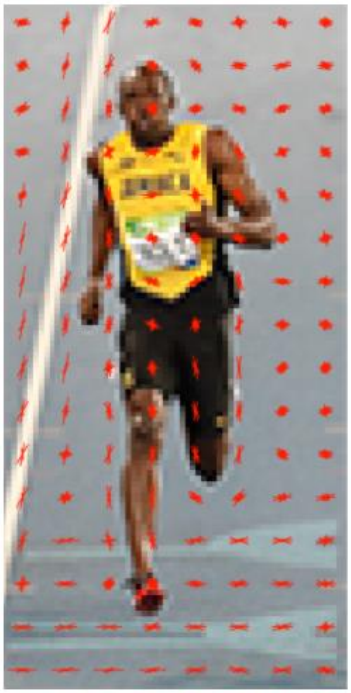
На попередньому кроці створено гістограму на основі градієнта зображення. Градієнти зображення чутливі до загального освітлення. Якщо зробити зображення темнішим, поділивши всі значення пікселів на 2, величина градієнта зміниться вдвічі, а, отже, значення гістограми зміняться вдвічі. В ідеалі – дескриптор не повинен залежати від змін освітлення. Іншими словами, необхідно «нормалізувати» гістограму, щоб на неї не впливали зміни освітлення. Нормалізований вектор гістограми отримується звичайним чином – кожен елемент вектора гістограми необхідно поділити на довжину вектора. Це називається L_2 – нормою.

Автори алгоритму не нормували вектор гістограми 9×1. Виявляється, що кращою ідеєю є нормалізація блоків більшого розміру – 16×16. Блок 16×16 має 4 гістограми, які можна об’єднати, щоб сформувати вектор елемента 36×1. Потім виділене вікно 16×16 переміщується на 8 пікселів, над цим вікном обчислюється нормалізований вектор 36×1 і процес повторюється.

Крок 5. Обчислення вектора функцій HOG

Для обчислення остаточного вектора ознак для всього вибраного фрагмента зображення, вектори 36×1 об'єднують в один вектор. Обчислимо розмір цього вектора. Є 7 горизонтальних та 15 вертикальних положень блоків 16 × 16, що в цілому становить $7 \times 15 = 105$ позицій.

Кожен блок 16×16 представлений вектором 36×1. Отже, після об'єднання в один вектор, отримуємо вектор, який складається з $36 \times 105 = 3780$ елементів.



Візуалізація гістограми орієнтованих градієнтів

Дескриптор HOG обраного фрагменту зображення зазвичай візуалізується шляхом побудови нормалізованих гістограм 9×1 у клітинках 8×8. Очевидно, що напрямок гістограми фіксує форму людини, особливо, навколо тулуба та ніг.

4.5. Опис програмного продукту

Варто відмітити, що часто реальні дані є дуже зашумленими та неоднорідними, їхню структуру та основні ознаки важко визначити внаслідок пропущених характеристик. Тому перш ніж застосувати будь-який з методів машинного навчання необхідно провести попередню обробку даних та визначити основні характеристики. Не існує універсальної формули, щоб зробити попередні кроки. Тому кожен спеціаліст повинен покладатися на власну інтуїцію та знання у певній галузі застосування. Класифікація зображень проводиться за допомогою функцій, змінними яких виступають пікселі. Розглянемо алгоритм HOG, який перетворює пікселі у векторне зображення. Алгоритм HOG є чутливим до різних інформативних характеристик зображення та не залежить факторів освітленості.

Варто відзначити, що даний пункт містить основні пояснення та підходи у реалізації програмного коду. Весь лістинг програми міститься в Додатку А.

Алгоритм виявлення ознак HOG вбудований в пакет Scikit-Image. Розглянемо приклад візуалізації методу орієнтованих градієнтів [20]:

```
from skimage import data, color, feature
import skimage.data

figure = color.rgb2gray(data.chelsea())
hog_vec, hog_vis = feature.hog(figure, visualise=True)

fig, cat = plt.subplots(1, 2, figsize=(12, 6),
                    subplot_kw=dict(xticks=[], yticks=[]))
cat[0].imshow(figure, cmap='gray')
cat[0].set_title('Вхідне зображення')
cat[1].imshow(hog_vis)
cat[1].set_title('Візуалізація ознак HOG')
```

Для навчання моделі спочатку необхідно отримати навчальну вибірку (позитивну) з бази даних Wild (міститься в Scickit-Learn), яка містить 13233 зображення людських облич.

```
In:  from sklearn.datasets import fetch_lfw_people
     f = fetch_lfw_people()
     pos_pat = f.images
     print(pos_pat.shape)

Out: (13233, 62, 47)
```

Надрукуємо фотографії зображень з позитивної навчальної вибірки. Ось так виглядають 40 фотографій:

```
fig, ax = plt.subplots(4, 10)
for i, axi in enumerate(ax.flat):
   axi.imshow(pos_pat[50 * i], cmap='gray')
```

```
    axi.axis('off')
```

60 фотографій з позитивної навчальної вибірки:

```
fig, ax = plt.subplots(6, 10)
for i, axi in enumerate(ax.flat):
    axi.imshow(pos_pat[60 * i], cmap='gray')
    axi.axis('off')
```

З допомогою PatchExtractor із бібліотеки Scickit-Learn отримуємо 30000 зображень, які не містять обличчя людей (негативна навчальна вибірка).

```
In:  from skimage import data, transform
     fig_to_use = ['camera', 'text', 'coins', 'moon',
                   'page', 'clock', 'immunohistochemistry',
                   'chelsea', 'coffee', 'hubble_deep_field']
     figures = [color.rgb2gray(getattr(data, name)())
                for name in fig_to_use]


     from sklearn.feature_extraction.image import PatchExtractor
     def extract_patches(img, N, scale=1.0, patch_size=pos_pat[0].shape):
         ext_pat_size = tuple((scale * np.array(patch_size)).astype(int))
         extractor = PatchExtractor(patch_size=ext_pat_size,
                                    max_patches=N, random_state=0)
         patches = extractor.transform(img[np.newaxis])
         if scale != 1:
             patches = np.array([transform.resize(patch, patch_size)
                                 for patch in patches])
         return patches

     neg_pat = np.vstack([extract_patches(im, 1000, scale)
                          for im in figures for scale in [0.5, 1.0, 2.0]])
     print(neg_pat.shape)
```

```
Out: (30000, 62, 47)
```

Далі об’єднуємо позитивну та негативну вибірки, утворюючи набір даних для навчання, та обчислюємо вектор ознак HOG.

```
In:  from itertools import chain
     X_train = np.array([feature.hog(im)
                         for im in chain(pos_pat,
                                         neg_pat)])
     y_train = np.zeros(X_train.shape[0])
     y_train[:pos_pat.shape[0]] = 1

     # Розмір навчальної вибірки
     print(X_train.shape)
Out: (43233, 1215)
```

Далі використовуємо класифікатор SVM для класифікації патчів. Патч – це рамка, яка вказує на наявність обличчя на фото. Розглянемо Linear SVM з бібліотеки Scikit-Learn, оскільки в порівнянні з SVM він часто має краще масштабування для великої кількості зразків у випадку двовимірних даних. Для початку розглянемо класифікатор NB

```
In:  from sklearn.naive_bayes import GaussianNB
     from sklearn.cross_validation import cross_val_score
     print(cross_val_score(GaussianNB(), X_train, y_train))

Out: array([ 0.9408785 ,  0.8752342 ,  0.93976823])
```

Як видно, навіть простий наївний алгоритм Байєса дає точність 94%, що є досить хорошим результатом. За допомогою крос-валідації отримуємо усереднені показники по всій вибірці. Розглянемо далі класифікатор SVM із пошуком сітки за кількома варіантами параметра C (на 1, 2, 4, 8 пікселів):

```
In:  from sklearn.svm import LinearSVC
     from sklearn.grid_search import GridSearchCV
     grid = GridSearchCV(LinearSVC(), {'C': [1.0, 2.0, 4.0, 8.0]})
     grid.fit(X_train, y_train)
     print(grid.best_score_)
     print(grid.best_params_)

Out: 0.98667684407744083
     {'C': 4.0}
```

Отже, класифікатор SVM працює з точністю 98,7% з використанням 4 клітинок сітки. Отримаємо далі найкращий класифікатор:

```
In:  model = grid.best_estimator_
     model.fit(X_train, y_train)
```

```
Out: LinearSVC(C=4.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
```

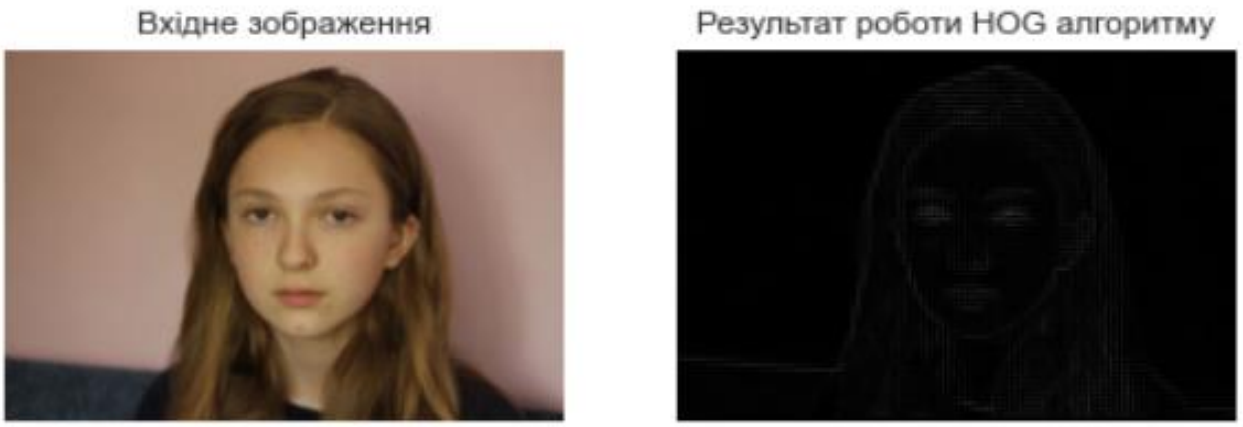
Далі розглянемо, як отримана модель працює з довільними вхідними зображеннями. Для цього спочатку потрібно написати функцію, яка перебирає набір клітинок на сітці (патчі) та обчислює HOG ознаки для кожного патчу.

```
In:  def sliding_window(img, patch_size=pos_pat[0].shape,
                        istep=2, jstep=2, scale=1.0):
         Ni, Nj = (int(scale * s) for s in patch_size)
         for i in range(0, img.shape[0] - Ni, istep):
             for j in range(0, img.shape[1] - Ni, jstep):
                 patch = img[i:i + Ni, j:j + Nj]
                 if scale != 1:
                     patch = transform.resize(patch, patch_size)
                 yield (i, j), patch

     indices, patches = zip(*sliding_window(testing))
     patches_hog = np.array([feature.hog(patch) for patch in patches])
     print(patches_hog.shape)
```

Подивимося на результат роботи алгоритму HOG для вхідних тестових зображень даної магістерської роботи:

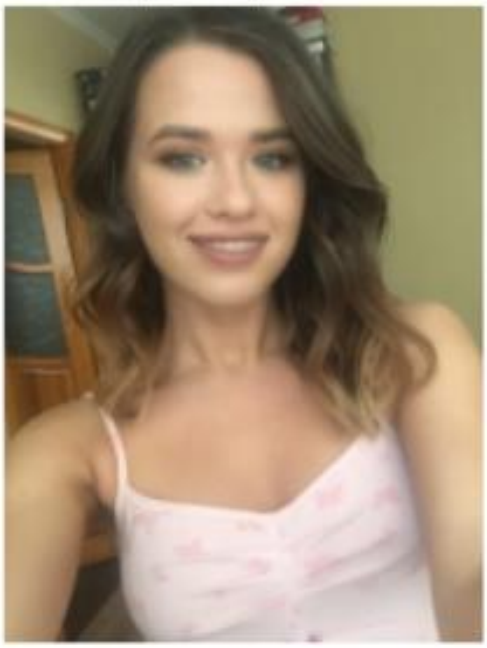
```
import matplotlib.pyplot as plt
from skimage.feature import hog
from skimage import data, exposure
import numpy as np
import matplotlib.pyplot as plt

image = np.array(Image.open('girl1.jpg'))

fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16),
                    cells_per_block=(1, 1), visualize=True, multichannel=True)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4), sharex=True, sharey=True)
```

```
ax1.axis('off')
ax1.imshow(image, cmap=plt.cm.gray)
ax1.set_title('Вхідне зображення')

hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

ax2.axis('off')
ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)
ax2.set_title('Результат роботи HOG алгоритму')
plt.show()

# Кожне фото має свій розмір за схемою RGB. Тому розмір фото є вектором з трьох елементів.
print(image.shape)
```




```
(1066, 1600, 3)
```

Вхідне зображення



Результат роботи HOG алгоритму

(1000, 750, 3)

Вхідне зображення

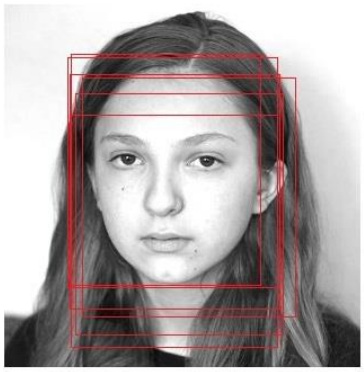
Результат роботи HOG алгоритму

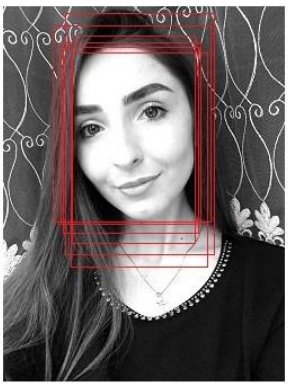
(1600, 1202, 3)

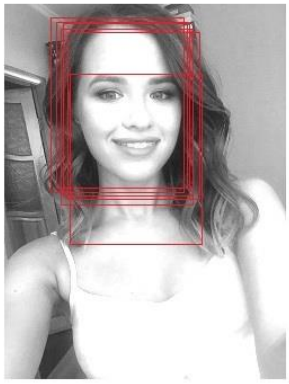

```
(1190, 893, 3)
```

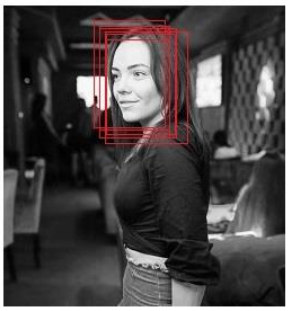

```
(630, 589, 3)
```

Нарешті, використовуючи всі патчі, що містять ознаки HOG, та нашу модель, можемо оцінити, чи кожен патч містить обличчя.









Смартфони, зазвичай, для одержання єдиної рамки для визначення обличчя на зображені усереднюють знайдені червоні прямокутники. Усереднений прямокутник зі скінченної кількості червоних прямокутників якраз і виводиться на екран, виділяючи обличчя.

А ось для того, щоб людське обличчя в масці також визначалось, необхідно, щоб навчальна позитивна вибірка містила такі ж зображення.

Висновки

У наш час важко переоцінити масштаби сфер застосування машинного навчання. Країни з найбільш розвиненими економіками світу борються за пальму першості у розробці нових підходів та методів дослідження у задачах розпізнавання об’єктів, класифікації та кластеризації даних. З допомогою машинного навчання стало можливим вивчити попередній досвід за вибраний період часу та отримати інформацію з даних, яка на перший погляд не була очевидною чи, навіть, гіпотетичною. Найперше, цей досвід є безцінним для медицини. Адже поруч із стрімким технічним прогресом, люди продовжують помирати навіть від нескладних захворювань. Саме діагностика знімків МРТ у поєднанні з методами розпізнавання ознак та класифікації даних може врятувати життя. Алгоритми машинного навчання порівнюють знімки МРТ з набором знімків тисяч людей. Таким чином стає зрозумілою найкраща методика лікування та прогнозування можливих ускладнень.

У даній роботі розглянуто покрокову реалізацію алгоритму розпізнавання ознак HOG. Алгоритм HOG є одним із найкращих алгоритмів, які добре працюють з фото даними, та не залежить від змін освітленості на фото. У якості навчальної вибірки вибрано базу даних Wild, яка міститься у Scikit-Learn. База даних Wild містить зображення облич людей. Для кращого процесу навчання моделі Wild доповнено зображеннями з SL бібліотеки, які не містять зображення людських облич. Тобто, навчальна вибірка складається з двох частин – позитивна (13233 зображень) та негативна (30000 зображень). Після виявлення ознак вхідного зображення з допомогою алгоритму HOG у дію вступає класифікатор SVM, який визначає, чи

присутнє на зображенні обличчя людини. Класифікатор SVM вибирає вказану кількість пікселів на зображенні, перебираючи таким чином все зображення. На виході отримується те ж вхідне зображення з червоними рамками (у випадку, якщо обличчя на фото присутнє).

Список використаних джерел

[1] https://www.hisour.com/ru/machine-learning-42773/

[2] https://doi.org/10.1093/mind/LIX.236.433

[3] https://habr.com/ru/company/toshibarus/blog/433544/

[4] Guyon, I., Elisseeff, A.: An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3, 1157–1182 (2003)

[5] Воронцов К. В.: Лекции по методам оценивания и выбора моделей (2007)

[6] Guyon, I., Elisseeff, A.: An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3, 1157–1182 (2003)

[7] Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)

[8] Kursa, M.B., Rudnicki, W.R.: Feature Selection with the Boruta Package. Journal Of Statistical Software 36(11) (2010)

[9] Tuv, E., Borisov, A., Runger, G., Torkkola, K.: Feature selection with ensembles, artificial variables, and redundancy elimination. The Journal of Machine Learning Research 10, 1341–1366 (2009)

[10] Laptin Yu., Likhovid A. P., Vinogradov A.P. Approaches to Construction of Linear Classifiers in the Case of Many Classes // Pattern Recognition and Image Analysis, Vol. 20, No. 2, 2010, p. 137-145.

[11] Рублев Б.В., Петунин Ю.И., Литвинко П.Г. Структура гомотетичных линейно разделимых множеств в n-мерном евклидовом пространстве. – Кибернетика и системный анализ. Ч.1 – 1992. – № 1. – С. 3-15; Ч.2 – 1992. – № 2. – С. 23-33.

[12] Platt J.C., Cristianini N., Shawe-Taylor J. Large margin DAG's for multiclass classification // Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2000, vol. 12, pp. 547–553.

[13] Журавлев Ю.И., Лаптин Ю.П., Виноградов А.П. Минимизация эмпирического риска и задачи построения линейных классификаторов // Кибернетика и системный анализ. 2011, № 4.- С. 155 – 164.

[14] Shor N. Z. Nondifferentiable Optimization and Polynomial Problems. – Amsterdam / Dordrecht / London: Kluwer Academic Publishers, 1998. – 381 p.

[15] Воронцов К.В. Машинное обучение. – http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_(курс_лекций%2C_К.В.Воронцов)

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition

(CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.

[17] https://www.learnopencv.com/histogram-of-oriented-gradients/

[18] https://www.digest.pro/news/preimushhestva-python-pered-drugimi-jazykami-programmirovanija/

[19] https://neurohive.io/ru/osnovy-data-science/vvedenie-v-scikit-learn/

[20] Jake VanderPlas. 2016. Python Data Science Handbook: Essential Tools for Working with Data (1st. ed.). O'Reilly Media, Inc.

[21] Fung G. M., Mangasarian O. L. Multicategory Proximal Support Vector Machine Classifiers // Machine Learning, 59, 2005, p. 77–97.

[22] Draminski, M., Rada-Iglesias, A., Enroth, S., Wadelius, C., Koronacki, J., Komorowski, J.: Monte Carlo feature selection for supervised classification. Bioinformatics 24(1), 110–117 (Nov 2008)

[23] Chih-Wei Hsu, Chih-Jen Lin A comparison of methods for multiclass support vector machines // IEEE Transactions on Neural Networks. – 2002, Volume 13, Issue: 2. – P**.** 415 - 425

Додаток А

```
from IPython import get_ipython
get_ipython().run_line_magic('matplotlib', 'inline')

import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np

# Приклад
from skimage import data, color, feature
import skimage.data

figure = color.rgb2gray(data.chelsea())
hog_vec, hog_vis = feature.hog(figure, visualise=True)



fig, cat = plt.subplots(1, 2, figsize=(12, 6),
                        subplot_kw=dict(xticks=[], yticks=[]))
cat[0].imshow(figure, cmap='gray')
cat[0].set_title('Вхідне зображення')

cat[1].imshow(hog_vis)
cat[1].set_title('Візуалізація ознак HOG');

# З бази даних Wild, яка доступна у бібліотеці Scikit-Learn, отримуємо навчальну вибірку,
яка містить фотографії облич людей (позитивні приклади)

from sklearn.datasets import fetch_lfw_people
f = fetch_lfw_people()
pos_pat = f.images
print(pos_pat.shape)

# Далі нам потрібні фото, на яких не зображені люди.
```

З допомогою PatchExtractor, який також доступний в бібліотеці Scikit-Learn, маємо ще одну навчальну вибірку (негативні приклади).

```
from skimage import data, transform

fig_to_use = ['camera', 'text', 'coins', 'moon',
              'page', 'clock', 'immunohistochemistry',
              'chelsea', 'coffee', 'hubble_deep_field']
figures = [color.rgb2gray(getattr(data, name)())
           for name in fig_to_use]



from sklearn.feature_extraction.image import PatchExtractor

def extract_patches(img, N, scale=1.0, patch_size=pos_pat[0].shape):
    ext_pat_size = tuple((scale * np.array(patch_size)).astype(int))
    extractor = PatchExtractor(patch_size=ext_pat_size,
                               max_patches=N, random_state=0)
    patches = extractor.transform(img[np.newaxis])
    if scale != 1:
        patches = np.array([transform.resize(patch, patch_size)
                            for patch in patches])
    return patches

neg_pat = np.vstack([extract_patches(im, 1000, scale)
                     for im in figures for scale in [0.5, 1.0, 2.0]])
print(neg_pat.shape)

# Подивимося на 40 зображень негативної вибірки
fig, ax = plt.subplots(4, 10)
for i, axi in enumerate(ax.flat):
    axi.imshow(neg_pat[500 * i], cmap='gray')
    axi.axis('off')
```

```
# Об'єднуємо вибірки та обчислюємо HOG функції
from itertools import chain
X_train = np.array([feature.hog(im)
                    for im in chain(pos_pat,
                                    neg_pat)])
y_train = np.zeros(X_train.shape[0])
y_train[:pos_pat.shape[0]] = 1

# Розмір навчальної вибірки
print(X_train.shape)



# НАВЧАННЯ МОДЕЛІ
# Використаємо для класифікації лінійний SVM (у бібліотеці LinearSVC - це функція LinearSVC)
# Для порівняння розглянемо спочатку Наївний Баєсовий класифікатор

from sklearn.naive_bayes import GaussianNB
from sklearn.cross_validation import cross_val_score

print(cross_val_score(GaussianNB(), X_train, y_train))

# LinearSVC

from sklearn.svm import LinearSVC
from sklearn.grid_search import GridSearchCV
grid = GridSearchCV(LinearSVC(), {'C': [1.0, 2.0, 4.0, 8.0]})
grid.fit(X_train, y_train)
print(grid.best_score_)
print(grid.best_params_)

# Обираємо найкращий класифікатор
```

```
model = grid.best_estimator_
print(model.fit(X_train, y_train))

# Тестування роботи HOG та LSVM
testing = skimage.data.load(boy, as_grey=False)
testing = skimage.color.rgb2gray(testing)
testing = skimage.transform.rescale(testing, 0.5)
testing = testing[:160, 40:180]


plt.imshow(testing, cmap='gray')
plt.axis('off');


# Функція для обчислення HOG для фрагментів зображення, які містять обличчя

def sliding_window(img, patch_size=pos_pat[0].shape,
                   istep=2, jstep=2, scale=1.0):
    Ta, To = (int(scale * k) for k in patch_size)
    for i in range(0, img.shape[0] - Ta, istep):
        for j in range(0, img.shape[1] - Ta, jstep):
            patch = img[i:i + Ta, j:j + To]
            if scale != 1:
                patch = transform.resize(patch, patch_size)
            yield (i, j), patch

indices, patches = zip(*sliding_window(testing))
patches_hog = np.array([feature.hog(patch) for patch in patches])
print(patches_hog.shape)


# Перевіряємо, скільки з них містить обличчя

labels = model.predict(patches_hog)
```

```
print(labels.sum())

# Перевіряємо результат

fig, ax = plt.subplots()
ax.imshow(testing, cmap='gray')
ax.axis('off')

Ta, To = pos_pat[0].shape
indices = np.array(indices)

for i, j in indices[labels == 1]:
    ax.add_patch(plt.Rectangle((j, i), To, Ta, edgecolor='red',
                               alpha=0.3, lw=2, facecolor='none'))
```