ОПИСАНИЕ НА АЛГОРИТЪМА ADABOOST, ИЗПОЛЗВАН ПРИ РАЗПОЗНАВАНЕТО НА ИЗОБРАЖЕНИЯ

Росен Капланов
Университет по библиотекознание и информационни технологии

Резюме: С бързо развиващите се технологии и разработването на изкуствен интелект лицевото разпознаване и разпознаването на изображения привличат все повече внимание.

В сравнение с традиционното разпознаване на карти, разпознаване на пръстови отпечатъци и разпознаване на ириса на ретината, разпознаването на лица има много предимства, като ограничение за безконтактност, висока съвместимост и удобство за потребителя. Също така има висок потенциал за използване от правителство, обществени услуги, сигурност, електронна търговия, търговия на дребно, образование и много други области.

В настоящия доклад ще бъде представен методът AdaBoost и ще бъдат разгледани основни техники при разпознаването на изображения и машинното обучение.

Ключови думи: AdaBoost, Algorithm, машинно обучение, разпознаване на изображения, pattern recognition.

Въведение

С развитието на методите за обучение и представянето на конволюционните невронни мрежи точността и бързината на лицевото разпознаване и разпознаването на изображения е постигат огромен прогрес. Резултатите от различните мрежи и модели на разпознаване могат да бъдат различни. Целта е да обединят определени модели и алгоритми с цел да се получат по-добри резултати, с по-голяма точност. След като е получен модел, който е с голяма точност, може да се премине към получаването на продуктовия (актуалния) модел на изображението.

Традиционната технология за разпознаване се основава на паметта на всеки индивид – потребителско име, парола, както и информация, като лични карти, дигитални ключове и др. Въпреки всичко рискът от придобиване на тези данни от други лица и обекти може да се превърне в сериозен проблем. Запазването на информацията и нейната оригиналност и интегритет е трудна задача, но още по-трудна е тази, при която информацията трябва да се предпази от чужда намеса. Иначе всеки индивид може да се представи

за друг и да придобие същата налична информация. Ако индивидуалността бъде придобита от други, това може да доведе до сериозни проблеми и последствия. Всеки ден обемът от информация, който се обработва, се увеличава в пъти и няма данни този процес да намалява. Обекти с огромни бази данни (военни формирования, медицина, производители на автомобили и др.) винаги са търсели автоматизиран начин да се съчетаят данни, като се използват различна логика и статистика. Смисълът на този доклад е да бъде анализиран комбиниран метод с помощта на AdaBoost (Adaptive Boosting).

1. Методология на изследването

Разпознаването на образи има своите корени в инженерството, докато машинното обучение се развива от компютърните науки.

Една от главните идеи за използването на машинното обучение е да се проектира компютърна система, базирана на разпознаването на образи, която ще има естествената възможност да се самообучава въз основа на наблюдение. Алгоритъмът не само ще изпълни тази задача, но също така ще предостави възможност на машината да се обучава от непълнотата на пакета от характеристики. Примери могат да бъдат търсачки, които подобряват показаните резултати, базирани само на техния опит (предишни заявки за търсене).

Един пълен процес на разпознаване на изображения се състои от три основни фази:

- намиране на лицето/изображението;
- извличане на характеристики на лицето/изображението;
- разпознаване на лицето/изображението.

Следователно е необходимо да се извлече областта на лицето от фаза обработка на изображението на лицето, а след това да се обработи и отдели лицето от фона, който осигурява основата за последващо извличане на характеристиките на лицето.

За да използваме метода, трябва да изградим система за разпознаване, като винаги за целта е нужно да имаме сетове от данни, за да можем да изградим определени категории и да открием приликите между тях във всяка категория. Тестовите данни обикновено се наричат querys (низове от заявки).

2. Машинно обучение и разпознаване на образи

„Pattern recognition“ [1] е специфична област в машинното обучение. Тя се фокусира по-дълбоко върху областта на възлагане на входния сигнал върху два или повече класа.

Намирането на начини да се комбинират различни методи и насоки, е решаваща стъпка в обработването на данни. В реалността няма перфектен pattern recognition модел, който върху всеки пакет от данни да превъзхожда останалите. Можем само да кажем следното: всеки модел превъзхожда друг и е по-добър от трети, ако се самообучава по-добре от другия или е по-добър класификатор от третия и т.н., повечето време.

Нека да разгледаме следния случай. Имаме една област от данни за изследване и различни класификатори – „Невронни мрежи“ (Neural Nets) [7], „Най-близкият съсед“ (Nearest Neighbor) [8], [11] „Бейсов мрежов класификатор“ (Bayesian Network Classifier) [1]. Когато всички те работят върху определен пакет или област от данни, ще доведат до различни резултати, като някои ще се справят по-добре и ще бъдат по-бързи от останалите.

3. Литературен обзор

Преди да вникнем по-дълбоко в pattern recognition стратегиите, трябва да отбележим различните вариации от комбинации, зависещи от типа на резултата от класификаторите. По този начин можем да разграничим три различни нива от резултатите от изчисленията:

– абстрактно ниво;
– ранг ниво;
– изчислително ниво.

Абстрактно ниво – при даден пакет от етикети Ω, пакет от класификатори D и пакет от обекти $a \epsilon R^n$, които да бъдат класифицирани. Всеки класификатор D_i поражда класове $S_i \epsilon \Omega, i = 1, \dots, L$, така че за всеки обект $a \epsilon R^n$ резултатът от изчисленията на класификатора дефинира вектор $s = [s_1 \dots, \dots, s_l]^T \epsilon \Omega^T$. При сравнение с другите подходи абстрактното ниво не ни доставя допълнителна информация за предварително предсказаните етикети, нито алтернативни предложения. Като пример може да бъдат дадени стратегии за асемблиране, като **Bootstrap aggregating** [1], известен като **Bagging**, и **Behavior** – **Knowledge Space**, те също използват абстрактно ниво на комбинаторика.

Ранг ниво – резултатът от всеки D_i е под пакет от Ω, където всеки класификатор е сортиран по ранг, като всеки е с по-голяма възможност да бъде точен, от другите. Като резултат от изчисленията се взема класификаторът с най-висок ранг. Ранг нивото е също познат метод като техника на гласуването. Смята се, че този метод е един от простите подходи за комбиниране на класификатори. Този подход е много подходящ, когато проблемът съдържа огромен брой

класификатори. Примери за ранг ниво подходи могат да бъдат Majority Vote, Borda count, Weighted Majority Vote [1].

Изчислително ниво – всеки класификатор поражда „С-измерим“ вектор $\left[D_{i,1} \ \ldots, D_{i,c}\right]^T$. Резултатната изчислена стойност $d_{i,j}$ е функция от входящите стойности на x и представлява хипотезата, че векторът x е предоставен за класифициране от клас ω_j. Проста комбинация от подходи за изчислително ниво прилага определена яснота и правила, като правилото на сумиране, продуктивност, осредняване, минимум и максимум, средна стойност.

4. Асемблирано обучаване

Асемблираното обучаване, познато още като супервайзорно обучаване, е подобно на метаалгоритъма, който използва много pattern recognition алгоритми като компонент за изучаване на колекция от предвиждания. Главната идея в методите на асемблиране е базирана на критерии от рода „Групата е по-умна от един-единствен член на групата“. Като се използват различни стратегии, се създават асемблиране и комбинация от предвиждания, които реферират към определен набор от алтернативни модели. За да разберем защо асемблирането ще превъзхожда резултата от един-единствен източник, ще посочим три причини, дефинирани от Дитрих [9]:

- В един малък пакет от данни алгоритмите за обучение могат да калкулират различни хипотези h(x), които ще дадат една и съща точност на изпитателните данни. С една конструкция от асемблиране можем някак си да нормализираме или осредним избора на един от грешните класификатори.
- Обучаващият алгоритъм може да приложи определен процес на търсене за най-добрата хипотеза в неправилен начин.
- При условие че правилната функция може и да не се съдържа в областта с хипотезите, асемблирането по критерий сумарна тежина ще ни даде много добър приблизителен резултат.
- Има различни видове стратегии за асемблиране. Ще се спрем по-подробно върху три от тях – Bagging, Boosting и Stacking.

4.1. Bagging

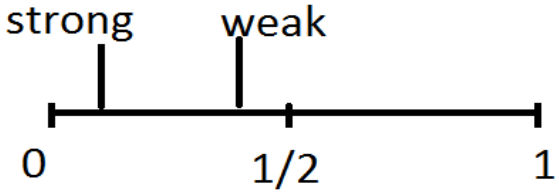
Методът Bagging е представен за първи път от Брайман през 1996 г. и е по-известен като bootstrap aggregating. Този метод е добре познат за генериране на ансамбъл от класификатори. Даден е един определен подбор от данни. Нека го наречем X. Bagging създава няколко нови тренировъчни подбора от данни X', всеки от които с размери X > X', вземайки произволни образци със заместване. По време на процеса някой от инстансите са репликирани, а други са

изтрити. Този процес продължава, докато всички новосъздадени тренировъчни подбори X' достигнат размерите на оригиналния: X = X'. По този начин, имайки различни вариации от данни, Bagging комбинира различните резултати от изчисления в едно-единствено предположение. За целите на добрата производителност е много важно разнообразието на генерираните данни. Изпитването на класификаторите върху два почти идентични подбора от данни ще доведе до два почти еднакви резултата, които няма да могат да подобрят крайния резултат от изчисления. Най-вероятната причина за проблемите с класификаторите е използването на вот (тежестта на вота). В случай на регресивни проблеми осредненият се използва за крайна хипотеза [3].

4.2. Boosting

Началната идея на Boosting страда от много недостатъци, които пораждат идеята на Шапир и Фрюнд за подобряване на алгоритъма чрез по-добра версия [9], [10].

Boosting за първи път е представен от Шапир и Фрюнд. Boosting използва определен подбор от слаби класификатори и се опитва да създаде модел на силен класификатор. Слабите класификатори са дефинирани с възможен процент за грешка, близък до 0,5. Силните класификатори са класификаторите с процент за грешка, близък до 0 [4].



Фиг. 1. Графика на силни и слаби класификатори

5. Резултати

AdaBoost се фокусира основно върху бинарната класификация на два класа, част от мултикласов проблем. Алгоритъмът използва като входящ поток поредицата (x_1, y_1) ... (x_n, y_n), където X представлява инстанс, а У – очаквания клас. За присвояването на двата класа стойността на У се счита да бъде в рамките $[-1:1]$. В началото тежестите на всички точки на данни са добавени. AdaBoost протича в серия от итерации R, където всяка итерация R извиква класификатор с най-малкия процент на грешка и обновява съответната тежест. След всеки кръг алгоритъмът се фокусира върху грешните класифицирани

инстанси, като подобрява тежестта (приоритета) им. Крайната хипотеза е продуцирана, вземайки знаковата стойност от сумата от тежестите на всеки инстанс (виж фиг. 1).

5.1. Stacking (Натрупване)

Stacking, познат също като Stacked Generalization, е още една техника за асемблиране, която се опитва да повиши точността на данните, базирайки се на тренировъчни подбори. Техниката обучава метакласификаторите, които се учат от базовите класификатори, или с други думи, използва като входящ поток предвижданията на всеки класификатор, а не оригиналния входен атрибут. Може също така да детерминира кои модели ще се представят добре и кои ще се провалят (позовавайки се на тежестта на резултата), и изключва онези, които не са рентабилни. Чрез комбинирането на различни метакласификатори финалният резултат е получен. Stacking много наподобява стратегията Boosting, но както беше казано, използва метаниво и обикновено се използва да комбинира модели от различен тип. За по-добро представяне се препоръчва данните да бъдат разделени на две части: първата да съдържа метаподбора от данни, а втората – класификатори на базово ниво.

Таблица 1. Предимства и недостатъци на трите стратегии за асемблиране

	Предимства	**Недостатъци**
BAGGING	Много подходящ за нестабилни модели, една малка промяна на опитния сет от данни може да доведе до голяма промяна в резултатите.	Bagging е труден за представяне от хората.
	Смятан за бърз и лесен за имплементиране.	След създаването на дървовидната структура последната е трудна за представяне.
	С богатата си колекция от променливи и голям подбор от данни, има възможност да подобри разнообразността си.	
	Детерминира коя от променливите може да допринесе за класифицирането.	

	Ефективно управлява липсващите сетове от данни.	
BOOSTING	Лесен за имплементиране.	Чуствителни данни.
	Лесен за имплементиране.	Ако слабите класификатори са сложни, методът е склонен да използва overfit и underfit, ако те са много слаби.
	Оценява най-добрия сет от класификатори и изключва останалите.	Представянето на алгоритъма зависи от данните на слабите класификатори.
	Опитът и знанията на слабите не са от значение.	Ако слабите класификатори са с малки отклонения и грешки, методът може да не е от полза.
	Бърз.	
	Обикновено не е засегнат от прекомерно наместване (overfitting).	
	Може да управлява многокласови проблеми.	
	Може да управлява многокласови проблеми.	
STACKING	Може да управлява класифицирани и регресивни данни.	Само MLR алгоритъмът е подходящ за ниво 1 генерализиране.
	Комбинира различни типове изчислени резултати.	Обикновено се използват възможностите на класовете отколкото техните предвиждания.
	Комбинира само най-подходящите класификатори.	
	Методът има възможност да увеличи и промени алгоритмите в мета- и базовите нива.	

6. Комбинаторни методи

Едно от основните предимства на методите за комбинаторика е, че повечето от тях не изискват предварително обучение. Това ни дава извода, че вотът е изпълнен независимо от представянето на всеки класификатор индивидуално. Прости стратегии като Минимум, Максимум, Осредненост, Форма на решение, тази част ще обясни Naïve Bayes.

6.1. Класификаторът Naïve Bayes

Naïve Bayes е смятан за прост класификатор, част от Bayesian мрежата. Базиран на възможните състояния, създава Naïve предположението, че всяко свойство е независимо от останалите, сравнявайки определено свойство на класа. Формулата на Bayes е дефинирана като:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \quad (1)$$

Тази формула може да бъде представена и като:

$$Posterior = \frac{prior*likehood}{evidence} \quad (2)$$

Posterior означава по-късен или краен – крайната вероятност, когато всички доказателства са събрани.

Финалната хипотеза е избрана от правилото на Posterior, което означава вероятностната променлива, обуславяща максимума на възможностите от един тренировъчен пример.

$$V_{max} = maxP(V_j) \prod P(ai|ji) \quad (3)$$

Naïve Bayes калкулира вероятностите от всеки възможен пример в един сет от данни. Поради тази причина, ако сетът от данни е много голям, това може да доведе до непредвидени проблеми. Калкулирането на всяка възможна вероятност може да доведе до използване на много време за изпълнение или може въобще да не е възможно изпълнението.

Предимства на Naïve Bayes:
- възможност да управлява класифициране на регресивни данни;
- прост за имплементация;
- не изисква допълнителни данни, ако Naïve Bayes притежава условно независимо предположение;
- дори да не притежава, е склонен да бъде изненадващо точен;
- бърз и ефикасен.

Недостатъци на Naïve Bayes:
- не се представя задоволително в регресивни проблеми;
- калкулирането на всички вероятности в огромни масиви от данни е трудно;

– малък сет от данни води до ниска точност;
– ако един определен клас и свойство не се появят в примерен сет от данни, най-честата вероятност е 0.

6.2. Класификаторът Majority Vote/Weighted Majority Vote

Majority Vote е смятан за една от най-известните и ясни техники. На основата на една проста стратегия резултатът от изчисленията е дефиниран от най-често използвания клас в един сет от данни. Обикновено трябва да бъде използван нечетен брой класификатори, за да бъде предотвратена ситуация на равенство, макар че, ако такава се създаде, някое от обичайните правила избира правилния [5].

Тежестта на мажоритарния вот е вариация на мажоритарен вот, като всяка една от всички тежести е прибавена към различен класификатор. Тежестта може да бъде получена, като се калкулира точността на всеки класификатор. Ако имаме даден сет от данни, ще бъдат взети резултатите от изчисленията на тези класификатори, които имат по-голяма тежест вместо на тези които са по точни, но имат по малка тежест.

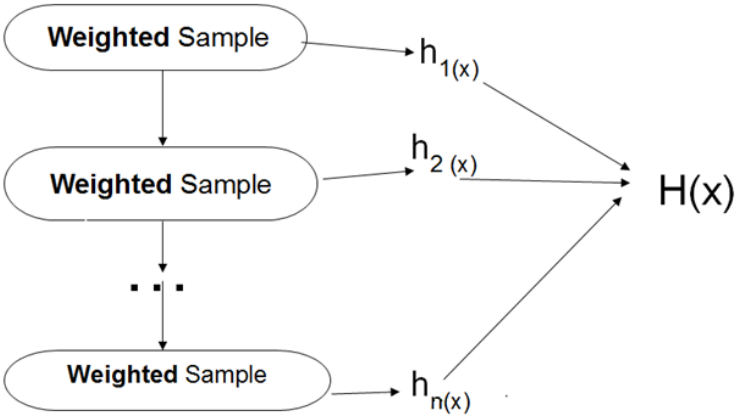
Предимства:
– много прост за имплементиране;
– с тенденция да бъде много точен, ако бъде приложен върху малки сетове от данни;
– резултатът може да бъде получен бързо;
– не е нужно предварително обучение.

Недостатъци:
– не може да управлява регресивни проблеми;
– точността зависи от разпределението на класовете;
– вътрешните връзки могат да доведат до произволни резултати;
– при мултикласови проблеми точността намалява.

7. Алгоритъмът за класификация AdaBoost

AdaBoost е машинен обучаващ метаалгоритъм, който се опитва да комбинира резултатите от изчисленията на многобройни слаби класификатори и да произведе един нов силен класификатор. Алгоритъмът използва базирано обучение, като създава сет от слаби класификатори, инициализирайки на всеки еднаква тежест и равнопоставеност. Класификаторът с най-малък процент за грешка е избран и неговата тежест е уеднаквена сред останалите. Алгоритъмът изпълнява рекурсивно определен брой от итерации, като във всяка избира класификатора с най-малкия процент за грешка и неговата

тежест. Класификаторите от предишните итерации се използват за манипулация на данните от предишните. Във всеки цикъл алгоритъмът се фокусира върху тези проби, които са дали грешка, като увеличава тяхната тежест. Финалният класификатор е произведен, като се взема знакът на сумата от тежестите на всяка комбинация, от всеки индивидуален класификатор, който е въвлечен в процедурата. Критичната стъпка на AdaBoost е изборът на класификатори с процент за грешка, по-малък от 0,5, на базата на тяхната тежест. Тези, които имат най-висок процент, няма да бъдат включени в калкулацията, тъй като алгоритъмът комбинира само класификаторите, които са рентабилни и са дали точен резултат от изчисленията (виж фиг. 2) [10].

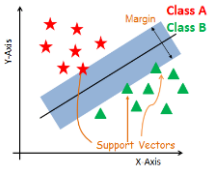


Фиг. 2. Идеята на алгоритъма AdaBoost

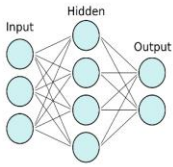
7.1. Пример за AdaBoost

Най-добрият начин да бъде обяснен алгоритъмът, е да се представи илюстриран пример. Нека да си представим следния примерен сет от данни, който е даден в таблицата. Таблицата съдържа данни от пет различни класификатора: Support Vector Machine (SVM) (фиг. 3), Neural Network (NN) (фиг. 4), Bayesian Network (BN) (фиг. 5), K-nearest Neighbor (KN) (фиг. 6) и SVM (N1). D1 представлява нашият очакван резултат от изчисления.

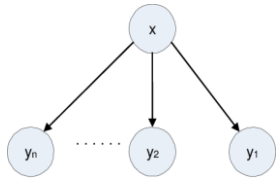
Целта на алгоритъма е да комбинира резултатите от този сет от класификатори и да представи резултат, който е близък до D1. Сравнявайки резултата от всеки класификатор с D1, можем да оценим цялостно точността на всеки индивидуален класификатор и по този начин да определим слабите класификатори (виж таблица 2).



Фиг. 3. Support Vector Machine (SVM)



Фиг. 4. Neural Network (NN)



Фиг. 5. Bayesian Network (BN)

Таблица 2. Примерен сет от данни

Point	SVM	BN	KN	SVM (N1)	NN	D1
A	1	1	0	1	0	1
B	1	0	0	0	1	1
C	0	1	1	1	1	1
D	1	0	1	0	0	1
E	1	1	1	1	1	1

P1(SVM | W) = 1/5
P2(NN | W) = 2/5
P3(BN | W) = 2/5
P4(KN | W) = 2/5
P5(SVM(N1) | W) = 2/5

P1 – P5 ни дава стойността на цялостната оценка на вероятностите за всеки класификатор, който е грешил спрямо дадения резултат в D1.

След като базовият класификатор е изпълнил теста си и слабите са създадени, изпълнението на алгоритъма може да започне. Първата стъпка на AdaBoost е да инициализира равнопоставеност на тежестите на всяка точка. Тежестта на всяка проба е калкулирана по формулата:

$$W_i = \frac{1}{Number\ of\ points} \qquad (4)$$

В следващия пример имаме пет data points, представени като A, B, C, D и E, които ще ни доведат до всяка data point, която има тежест 1/5.

Таблица 3. Тегловни индекси за всяка точка

Point	Round 1
WA	1/5
WB	1/5
WC	1/5
WD	1/5
WE	1/5

След това алгоритъмът избира класификатора с най-малкия процент, базирайки се на тежестите, по-малки от 0,5. В първия цикъл от алгоритъма AdaBoost ще избере класификатора SVM поради това, че поражда грешка само в data point (C) и по този начин завършва с процент за грешка, равен на 1/5 (виж таблица 3).

Следващата важна стъпка е да се калкулира алфа α. Алфа представлява реалната тежест, която е зададена на избрания класификатор, и е използвана за калкулиране на финалната хипотеза. Тази стойност може да бъде изчислена със следната формула:

$$\alpha = 0{,}5ln\left(\frac{1-error\ rate}{error\ rate}\right) \qquad (5)$$

Можем лесно да изчислим, че SVM ще има стойност на α = 0,5ln(4). Това слага край на първия цикъл. Ако на потребителя е нужна само една итерация, работата на алгоритъма приключва тук. Финалната хипотеза H(x) ще бъде изградена директно върху резултата от изчисленията от SVM респективно за всяка точка. Най-накрая ще приключим с точност 80%, която не е толкова лоша спрямо дадения сет от данни, но за смисъла да се използва AdaBoost, не е подходящо да има само една итерация.

Първата стъпка от втория цикъл е да бъдат обновени тежестите на всеки от класификаторите. Както вече беше описано, алгоритъмът обработва коректните и некоректните точки различно. Тежестите на коректно класифицираните точки са калкулирани по формулата:

$$W_{correct} = 0{,}5\left(\frac{OldWeght}{1-Error\ rate}\right) \qquad (6)$$

Тежестите на некоректните са калкулирани респективно по формулата:

$$W_{wrong} = 0{,}5\left(\frac{OldWeight}{Error\ rate}\right) \qquad (7)$$

Заключение

Този метод може да бъде предложен за използване при много проблеми, свързани с машинното обучение, като дава практически решения. Също така AdaBoost е много прост и интелигентен алгоритъм, има малки недостатъци и не може да бъде използван като единствено решение за класифициране на някои сетове от данни.

Настоящият доклад описва част от работата по дисертация. Предстои да се разгледат още проблеми и да се намери решение. Някои от тях са:

- разширяване на AdaBoost Algorithm за мултикласови проблеми;
- интелигентно разграничаване между данните от класификаторите и очакваните резултати;
- включване на идеални силни класификатори в AdaBoost изчисленията, маскирайки тяхното ниво на грешка от 0 с например 0,000001;
- имплементиране на повече интелигентни алгоритми, които могат да работят с регресивни проблеми.

References/Литература

1. **Bishop**, C. Pattern Recognition and Machine learning (October 2007), https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf
2. **Theodoridis**, S. Pattern Recognition, Fourth Edition (November 3, 2008), https://darmanto.akakom.ac.id/pengenalanpola/Pattern%20Recognition%204th%20Ed.%20(2009).pdf
3. **Kuncheva**, L. Combining Pattern Classifiers: Methods and Algorithms, (July 1, 2004), http://www.ccas.ru/voron/download/books/machlearn/kuncheva04combining.pdf
4. **Freund**, Y. Boosting: Foundations and Algorithms (January 10, 2014), https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Boosting_%20Foundations%20and%20Algorithms%20%5BSchapire%20%26%20Freund%202012-05-18%5D.pdf
5. **Kuncheva**, L., M. **Skurichina**, W. P. W. **Duin**. An experimental study on diversity for bagging and boosting with linear classifiers, http://homepage.tudelft.nl/a9p19/papers/fusion_03_diversity.pdf
6. **A Basic** Introduction To Neural Networks, https://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

7. **k-nearest** neighbours,
https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html
8. **Dietterich**, T. Ensemble Methods in Machine Learning,
https://web.engr.oregonstate.edu/~tgd/publications/mcs-ensembles.pdf
9. **Schapire**, R. – Explaining AdaBoost –
http://rob.schapire.net/papers/explaining-adaboost.pdf
10. **Elkan**, C. Nearest Neighbor Classification (2010),
https://cseweb.ucsd.edu/~elkan/250Bwinter2010/nearestn.pdf

За автора

Росен Капланов е докторант в УНИБИТ по програма „Автоматизирани системи за обработка на информация и управление на данните“.

Завършил е бакалавърска степен в ТУ – София през 2011 г. – електроинженер, специалност „Проектиране и изграждане на електрически апарати и системи“.

Завършил е магистърска степен в НБУ – София през 2015 г. – специалност „Софтуерни технологии в Интернет“.

Понастоящем е служител на ИБМ България – Отдел за проектиране и поддръжка на среди за програмиране на COBOL, PL/I, XL C++, Python.

За връзка с автора: rosen.kaplanov@gmail.com

OVERVIEW OF ADABOOST ALGORITHM USED IN IMAGE RECOGNITION TECHNOLOGIES

Rosen Kaplanov
University of Library Studies and Information Technologies

Abstract: With rapidly evolving technologies and development of artificial intelligence, facial recognition and image recognition are gaining more and more attention.

Compared to traditional card recognition, fingerprint recognition and retinal iris recognition, face recognition has many advantages, such as non-contact limitation, high compatibility, and user-friendliness. It also has high potential for use by government, public services, security, e-commerce, retail, education, and many other areas.

In-depth training in the field of machine learning is one of the new important areas that increasingly engage resources for development and improvement. This paper will present the AdaBoost method and discuss the basic techniques for image recognition and machine learning.

Keywords: AdaBoost, Algorithm, Machine Learning, Face Recognition.

About the author
Rosen Kaplanov
Bachelor's degree in Electrical Engineering
Master's Degree in Software Technologies
Employee of IBM in IBM Compilers department, supporting software environments for COBOL, PL/I, Python for z/OS, XL C++ for z/OS.
To contact the author: rosen.kaplanov@gmail.com