Годишник на секция “Информатика”
Съюз на учените в България
Том 12, 2022-2023, 72-94

Annual of “Informatics” Section
Union of Scientists in Bulgaria
Volume 12, 2022-2023, 72-94

Метод за извеждане в машинното самообучение, основан на метода на Кемени–Йънг

Александър Маразов

Секция “Биоинформатика и математическо моделиране”,
Институт по биофизика и биомедицинско инженерство,
Българска академия на науките
ул. “Акад. Г. Бончев”, блок 105, София 1113, България
E-mail: alexander@biomed.bas.bg

Резюме: Разглеждаме нови начини за извеждане, базирани на интеркритериалния анализ и метода на Kemeny–Young в машинното самообучение. Показваме приложение при диагностиката на болестта на Алцхаймер. Изложен е практичен метод за провеждане на изчисленията, въпреки NP трудността на метода. Предложеният метод за интекритериално извеждане дава консистентни резултати, които по начина на построение са устойчиви на грешки.
Ключови думи: Интеркритериален анализ, Метод на Kemeny–Young, Извеждане.

1 Увод

В тази работа ще предложим нови начини за извеждане, базирани на метода на Кемени–Йънг (Kemeny–-Young) в машинното самообучение. Подходът разширява предишни резултати на автора.

2 Методът на Кемени–Йънг

Този метод за агрегация на изборни резултати е предложен от Джон Кемени [8] през 1959 г., а по-късно Пейтън Йънг през 1988 г. показа, че той е оценка с максимална вероятност за предпочитания на двойки относно ранговете при предположението, че избирателят случайно ще размени двама кандидати от истинския ранг при $p < 0.5$ [17, 7].

Методът на Кемени–Йънг (Kemeny-–Young)[16] е математически метод за определяне на консенсус в множество от взаимно свързани предпочитания, посредством намирането на

рангове или подредби, които най-добре отразяват общата воля на участниците. Този метод е особено приложим в контекста на гласувания, анкети и вземане на решения, където се изисква обединяване на предпочитанията на група индивиди или експерти.

Идеята зад метода на Кемени–Йънг е да се определи рангова подредба на алтернативите, която минимизира разликата между тази подредба и индивидуалните предпочитания на всеки участник. За целта се използва матрица на предпочитания, която отразява степента на предпочитане на всяка двойка алтернативи от страна на всеки участник. Тази матрица се използва за съставяне на алтернативни рангове, които след това се сравняват с цел определяне на общата рангова подредба.

Методът на Кемени–Йънг представлява математическа техника, която може да бъде приложена в различни области, в които се изисква агрегиране на предпочитанията на група участници или експерти.

Предполагаме, че имаме множество от n алтернативи $X = \{x_1, x_2, ..., x_n\}$ и матрица на предпочитания P, където P_{ij} означава предпочитането на алтернатива x_i пред алтернатива x_j. Тези предпочитания могат да бъдат измерени, например, чрез брой гласове за дадена двойка алтернативи.

Целта е да се намери рангова подредба σ на алтернативите, която минимизира сумата на разликите между предпочитанията на участниците и ранговата подредба:

$$\min \sum_{i<j} P_{\sigma(i)\sigma(j)}$$

при условие, че σ е пермутация на индексите $\{1, 2, ..., n\}$.

Това може да бъде формулирано като задача за намиране на минимален хамилтонов цикъл, което е NP-трудна задача [13]. Нека всяка връзка има тегло, което е свързано с предпочитанията между две алтернативи. За практическото решаване на такива задачи се използват оптимизатори за целочислени задачи (MIP solver)[13].

3 Интуиционистки размити интерпретации на метода на Кемени–Йънг

Интеркритериалният анализ дава възможност да се даде интерпретация в термините на интуиционистки размити множества на метода на Кемени–Йънг. Степените на неопределеност и непринадлежност в интуиционистки размитите множества са удобен инструмент за оценка на качеството на предвидените резултати.

Методът на Кемени–Йънг използва бюлетини, върху които избирателите класират вариантите според техния ред на предпочитание. Избирател може да класира повече от един вариант на едно и също ниво на предпочитание. Некласирани варианти обикновено се тълкуват като най-малко предпочитани. За да установим съответствие с термините на интеркритериалния анализ ще използваме следните правила:

- равнокласифицираните варианти ще увеличават брояча на неопределеност
- предпочитанията ще увеличават брояча на принадлежност

Ще въведем следните символи, за по-удобни означения:

Дефиниция 1. С $X \prec Y$ означаваме, че Y е предпочитан пред X.

Дефиниция 2. С $X \preceq Y$ означаваме, че Y е предпочитан пред X или X и Y са еднакво предпочитани.

Дефиниция 3. С $X \succ Y$ означаваме, че Y е предпочитан пред X.

Дефиниция 4. С $X \succeq Y$ означаваме, че X е предпочитан пред Y или X и Y са еднакво предпочитани.

Дефиниция 5. С $X \sim Y$ означаваме, че X и Y са еднакво предпочитани.

Отбелязваме, че в случая $X \sim Y$ имаме едновременно $X \succeq Y$ и $X \preceq Y$. Изчисленията по метода на Кемени–Йънг обикновено се извършват в две стъпки. Първата стъпка е създаването на матрица или таблица, която брои двустранните предпочитания на избирателите. Втората стъпка е да се тестват всички възможни класирания, да се изчисли оценка за всяко такова класиране и да се сравнят оценките. Всяка оценка за класиране е равна на сумата от двустранните предпочитания, които се отнасят за това класиране.

Класирането, което има най-голяма оценка, е избраното по метода на Кемени–Йънг. Ако повече от едно класиране има еднаква оценка, избираме едно от тях на произволен принцип.

Друг еквивалентен начин за разглеждане на класирането е този, който минимизира сумата от разстоянията на Кендал (разстоянието от сортировката по метода на мехурчето).

За да видим как можем да направим интерпретация в термините на интуиционистки размитите множества на метода на Кемени–Йънг, ще разгледаме следния пример, адаптиран от [15]. Нека разгледаме следните предпочитания за четирима кандидати Елена, Румен, Мария и Симеон. Първо предпочитание: Елена, второ: Румен, трето: Мария или Симеон (равно предпочитание).

Възможни двойки от имена на избори	**Предпочитания Предпочитат X пред Y**	**Равнопоставени предпочитания**	**Предпочитания Предпочитат Y пред X**
X = Симеон, Y = Мария	0	+1 глас	0
X = Симеон, Y = Елена	0	0	+1 глас
X = Симеон, Y = Румен	0	0	+1 глас
X = Мария, Y = Елена	0	0	+1 глас
X = Мария, Y = Румен	0	0	+1 глас
X = Елена, Y = Румен	+1 глас	0	0

Таблица 1: Примерна таблица с броене на предпочитанията на един гласоподавател

3.1 Броене и класиране по метода на Кемени–Йънг

Нека въведем формално дефиниция за класиране.

Дефиниция 6. Класиране наричаме наредена n-орка $(X_1, X_2, \ldots, X_n)$, подредена от най-ниско класирания елемент X_1 до най-високо класирания елемент X_n.

В общия случай не съществува класиране $(X_1, X_2, \ldots, X_n)$, което за всеки $i < j$ всеки избирател $X_i \preceq X_j$. Това мотивира нуждата от процедура, която да даде класиране, което е оптимално в някакъв смисъл.

Продължаваме да разглеждаме примера за избора на четиримата кандидати Елена, Румен, Мария и Симеон. Сто човека дали предпочитанията си за тези четири кандидати. След като всички бюлетини са преброени, таблица може да бъде използвана, за да се съберат всички предпочитания на всички избиратели. Ето един пример за случай, в който са събрани гласовете на 100 избиратели, всеки от който има таблица, подобна на Таблица 1:

Възможни двойки от имена на избори	**Предпочитания Предпочитат X пред Y**	**Равнопоставени предпочитания**	**Предпочитания Предпочитат Y пред X**
X = Симеон, Y = Мария	50	10	40
X = Симеон, Y = Елена	40	0	60
X = Симеон, Y = Румен	40	0	60
X = Мария, Y = Елена	40	0	60
X = Мария, Y = Румен	30	0	70
X = Елена, Y = Румен	30	0	70

Таблица 2: Таблица с броене на предпочитанията от сто избиратели

Сумата от броя гласове във всяка редица трябва да е равна на общия брой гласове.

След като таблицата с броене е попълнена, всяко възможно класиране на изборите се разглежда поотделно, и оценката му за класиране се изчислява, като се добавя подходящото число от всяка редица на таблицата с броене. Например, за възможното класиране:

(Симеон, Мария, Румен, Елена)

удовлетворяващо предпочитанията
Елена $\succ$ Румен, Елена $\succ$ Мария, Елена $\succ$ Симеон, Румен $\succ$ Мария, Румен $\succ$ Симеон и Мария $\succ$ Симеон, съответните оценки, взети от таблицата, са:
Елена $\succ$ Румен: 30
Елена $\succ$ Мария: 60
Елена $\succ$ Симеон: 60
Румен $\succ$ Мария: 70
Румен $\succ$ Симеон: 60
Мария $\succ$ Симеон: 40
давайки обща оценка за класиране от $30 + 60 + 60 + 70 + 60 + 40 = 320$.

В контекста на интеркритериалния анализ, това число представлява сумата от степените на принадлежност S^{μ}_{ij} за всички двойки различни кандидати i, j.

S^{ν}_{ij} е сумата от несъгласните с подредбата i, j. S^{π}_{ij} е сумата на гласовете, които не предпочитат нито един от кандидатите i, j. Методът на Кемени–Йънг игнорира това

число. По-долу ще разгледаме използването на степен на неопределеност за получаване на характеристика за надеждността на оптималната подредба.

В контекста на ИРМ, Таблица 2 може да се интерпретира като степен на принадлежност, неопределеност и непринадлежност на предпочитанието $X \preceq Y$, за съответните стълбове след нормализация.

3.2 Изчисляване на общото класиране

След като се обходят всички възможни пермутации на класиранията на кандидатите и се изчисли оценка за всяко, може да се идентифицира класирането, което има най-голяма оценка, и това става общото класиране според метода на Кемени–Йънг.

В разгледания пример общото класиране с най-добър резултат е

(Мария, Симеон, Елена, Румен)

с оценка за класиране от 370.

3.3 Матрица на обобщението

След като общото класиране е изчислено, броят на сравненията на двойки може да бъде организиран в матрица на обобщението, както е показано по-долу. Тази матрична организация не включва равнопоставените сравнения на двойки, които се появяват в таблицата с броене:

	... пред Румен	**... пред Елена**	**... пред Симеон**	**... пред Мария**
Предпочитат Румен	-	70	60	70
Предпочитат Елена	30	-	60	60
Предпочитат Симеон	40	40	-	50
Предпочитат Мария	30	40	40	-

Таблица 3: Матрица на обобщението за общото класиране получено по метода на Кемени–Йънг: (Мария, Симеон, Елена, Румен). Числата над главния диагонал показват броя на съгласните с класирането. Числата под главния диагонал показват броя на несъгласните с класирането.

4 Извеждане в машинното самообучение

Извеждане в машинното самообучение е процесът на получаване на резултат или предсказания от модела, който е бил обучен с помощта на данните. След като моделът е обучен чрез подходящ алгоритъм върху обучаващия набор от данни, извеждането се използва за получаване на предсказания за нови или непознати данни, които не са били част от обучението.

Процесът на извеждане включва подаване на входни данни към обучения модел и извличане на отговор или класификация, които моделът предсказва за тези данни. Това

може да включва различни видове предсказания, като например класификация на обекти, регресия за предсказване на непрекъснати стойности или генериране на текст.

Важно е да се отбележи, че извеждането не включва обновяване на параметрите на модела. Те са фиксирани след обучението, а извеждането използва тези параметри за извеждане на предсказания. Това прави извеждането по-бърз от обучението, тъй като не се изисква промяна на параметрите на модела.

В областта на машинното самообучение извеждането играе ключова роля, тъй като предоставя възможност за използване на обучените модели за решаване на реални проблеми и приложения, като например предсказване на тенденции в данните, автоматизирано класифициране на обекти или анализ на текст.

Класификационната задача в машинното самообучение е процесът на присвояване на категория или клас на нови обекти въз основа на техните характеристики или признаци. Тази задача се използва, когато трябва да се предскаже към кой от фиксиран набор от възможни класове или категории принадлежи даден обект.

Примери за класификационни задачи включват:

- Медицинска диагностика: Класифициране на пациентите според наличието или липсата на определена болест или състояние.
- Идентификация на изображения: Категоризиране на изображения според тяхното съдържание, като например разпознаване на животни, обекти или лица.

5 Проблеми при извеждането на класификационни задачи с няколко класа

Нека разгледаме обобщението на задачата за класифициране от два класа към K-класа. Проблемите при класификацията на множество класове могат да представят специфични предизвикателства, които се различават от тези в двоичните класификационни задачи. Класификаторите могат да произведат -мерен вектор със стойности, съответстващи на сигурността за всеки клас. Освен това, има два други подхода за справяне с множествената класификация: “един срещу един” (OvO) и “един срещу останалите” (OvR), всеки със своите предимства и ограничения.

Глава 9 от [5] обхваща методите за многокласова класификация, включително OvO и OvR, като предоставя изчерпателен преглед на техните принципи и практически аспекти.

5.1 Един срещу един (OvO):

В OvO (One vs One) се обучава двоичен класификатор за всяка двойка класове. За N класа това дава K(K-1)/2 класификатора. Въпреки че изисква повече класификатори, OvO може да бъде изчислително ефективен за алгоритми, които се мащабират добре с броя на образците Такъв пример е методът на опорните вектори (support vector machines, SVM). Въпреки това OvO може да страда от дисбаланс в разпределението на класовете в получените двоични данни.

5.2 Един срещу останалите (OvR):

В OvR (One vs Rest) се обучава двоичен класификатор за всеки клас, като се разглеждат екземплярите на този клас като положителни примери, а екземплярите на всички останали класове – като отрицателни примери. OvR обикновено дава N класификатора за N класа. Въпреки че OvR може по-ефективно да се справя с дисбаланса в класовете отколкото OvO, той може да доведе до по-малко точни резултати, ако класификаторите са насочени към преобладаващия клас.

Авторите в [12] сравняват предимствата на класификацията OvR и представят емпирични доказателства в подкрепа на нейната ефективност в сравнение с други стратегии.

Хсу и Лин сравняват в [6] различни стратегии за многокласови SVM, включително OvO и OvR, и предоставят сравнение в тяхното относително представяне и изчислителна ефективност.

5.3 Нормализирани вероятности

Невронните мрежи и други класификатори могат да дадат изход в многомерен вектор. Най-често при класификация се ползва метод за нормализиране на изходните вектори. При него за всеки пример генерираме вектор с дължина, равна на броя на класовете, чийто елементи се сумират до 1. Всеки елемент на вектора се интерпретира като "вероятност" класа на съответната позиция да е действителния клас. Вероятност е в кавички, защото без калибрация тези числа не са близки до истинските вероятности.

6 Калибрация на класификатори

Калибрацията в машинното обучение е процесът на коригиране на прогнозите или вероятностите, предоставени от моделите, така че те да бъдат възможно най-близки до реалните стойности или вероятности на събитията, които се опитваме да предскажем [18]. Този процес се извършва с цел увереността в модела да бъде максимизирана и да се подобри неговата точност.

Ще използваме следната дефиниция за калибрация, представена в [14]. Един K-класов вероятностен класификатор е добре калибриран, ако сред тестовите инстанции, получаващи предсказан K-размерен вектор на вероятности s, разпределението на класовете е (приблизително) разпределено като s. Формално, може да изразим това със следната дефиниция

Дефиниция 7. Нека f е класификатор, който на пример X, дава вероятностен вектор s. f калибриран, ако

$$P(Y = i|f(X) = s) = s_i$$

Тази дефиниция е по-строга от други, защото оценява не само класа победител, а и разпределението на класовете претенденти. Това означава, че класификаторът правилно оценява нивото на неопределеност или увереност, свързана с неговите прогнози за всички класове във всеки пример.

Нека илюстрираме тази дефиниция с пример от метеорологията. Когато за даден ден се твърди, че има 80% вероятност за дъжд, това не означава, че ще вали 80% от максималния възможен дъжд, нито че ще вали 80% от времето. Метеорологът дава оценка, че средно от 5 прогнози с 80% вероятност една ще е грешна. Нека разгледаме пример в Таблица 4, адаптирана от [4]. В първата колона имаме разпределението на вероятностите на модела и реализираните вероятности за дъжд. Този модел е добре калибриран, защото реализираните вероятности са близки до агрегационните интервали. При твърде уверен модел реализираните вероятности ще са под предвидените в интервала. Например, интервала 90-100 процента ще има значително под 90% от примерите. При твърде неуверен модел, реализираните вероятности ще са значително над предвидения интервал. Например, интервалът 40-49, ще има реализирана вероятност 80%.

Предвиждане(%)	Брой прогнози	Брой реален валеж	Реализиран (%)
Над 90	1	1	100
80 до 89	1	1	100
70 до 79	7	6	86
60 до 69	15	11	75
50 до 59	13	8	62
40 до 49	15	8	53
30 до 39	18	6	33
20 до 29	31	7	23
10 до 19	22	4	18
0 до 9	0	–	–

Таблица 4: Предвидена вероятност и реализирана вероятност за дъжд. Адаптирана от [4]. Дава пример за разпределението на вероятностите на модела и реализираните вероятности за дъжд. Този модел е добре калибриран, защото реализираните вероятности са близки до агрегационните интервали.

Множество алгоритми за машинно обучение са известни със създаването на прекомерно уверени модели, заради което е нужно да се прилагат специални процедури по време на обучението. Методите за калибриране се прилагат след като модела е трениран. Те използват валидационни данни, за да се състави функция за калибриране на предварително обучен модел, която трансформира прогнозите на модела, за да бъдат по-добре калибрирани. Много методи за калибриране за двоични класификатори са били представени:

- логистично калибриране (известно и като “Platt scaling”)
- изотонично калибриране (известно и като метод на ROC изпъкнала обвивка)
- бета калибриране и байесови методи [14]

За да се извърши калибрация, се използват различни методи и техники, като например калибриране чрез линейни преобразувания [14]. Тези методи коригират изходите на моделите, за да се съобразят с реалните вероятности на събитията, като се използва статистическа обработка на данните за оценка и корекция на прогнозите на модела [10]. Използва се валидационен дейтасет, който е предварително отделен за целта.

Важно е да се отбележи, че калибрацията не винаги е задължителен етап в процеса на машинно обучение, но може да бъде полезен, особено когато точните вероятности на събитията са от съществено значение, като например във финансови прогнози или медицински приложения.

Най-общо методите могат да се разделят на параметрични и непараметрични [14]. Параметричните намират параметрите на функция, която най-добре приближава предвижданията на модела до реализираните вероятности. Пример за параметричен метод е Бриер [2]. Непараметричните методи разглеждат само подредбата на предвидените класове. Изотоничната калибрация е пример за непараметричен метод [1].

Ще покажем как може да получим оценка за правдоподобността на резултата в термините на интуиционистки размитите множества. Това представлява непараметричен метод за калибрация, основан на метода на интеркритериалния анализ.

7 Агрегиране на резултатите

Новият подход за агрегация получаваме като вземем наредбите от различните натренирани класификатори и изберем победителя по метода на Кемени–Йънг. За да можем да ползваме новия подход за извеждане достатъчно бързо, ще използваме стандартна процедура за преобразуване на метода на Кемени–Йънг в оптимизационна задача. За вход на процедурата използваме различните класирания от класификаторите, които искаме да агрегираме. Процедурата е описана във функцията

aggregate_rank_lp

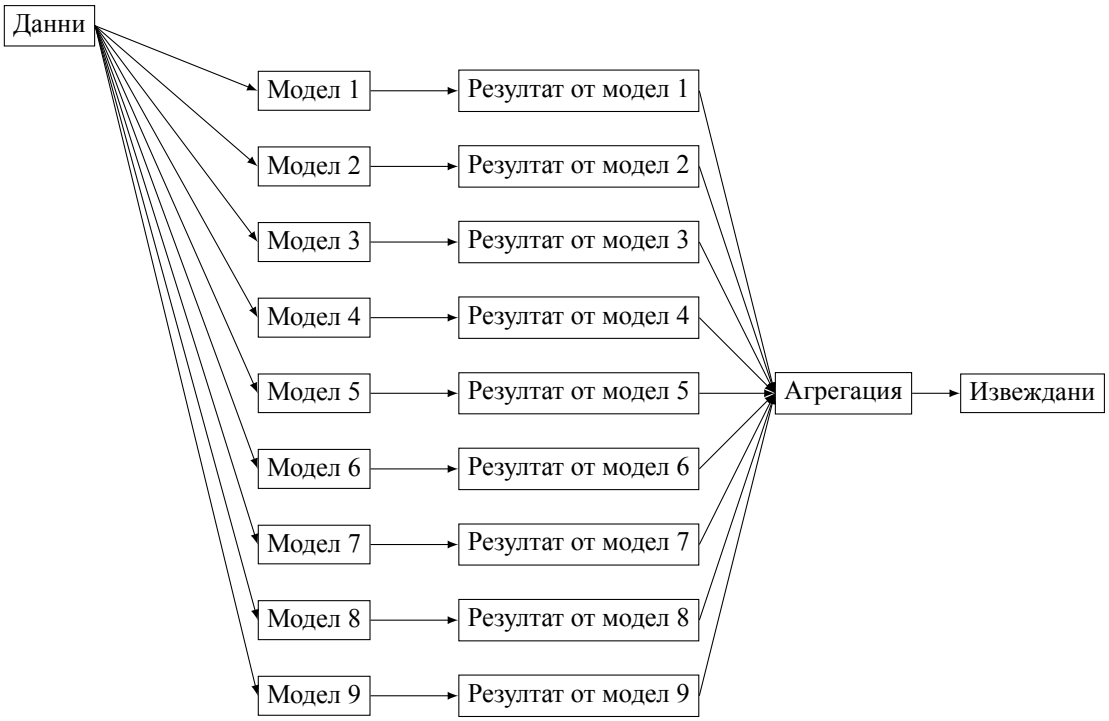
в Приложение Б. Изчислителната оптимизация на метода е описана в Раздел 8.

Оптимизационната задача формулираме като минимизиране по пермутациите на възможните класирания на сумата от двойките преференции,които не отговарят на разглежданото класиране. Добавят се ограничения върху променливите, така че да наложим транзитивност и асоциативност над полученото класиране.

7.1 Приложение

В [9] беше представен метод за диагностициране на болестта на Алцхаймер с невронни мрежи. Ще използваме същия подход, за да получим ансамбъл от класификатори, върху които ще приложим метода, описан в Раздел 7. На дейтасета с ЯМР изображения за диагностициране на болестта на Алцхаймер тренираме девет класификатора. За да получим по-добро разнообразие на предвидените резултати, използваме три различни архитектури: resnet18, resnet34 и resnet50, като за всяка тренираме три модела. Така получаваме деветте класификатора както е описано на Фигура 1. В [9] е използвана само архитектурата resnet34, но сега се стремим да разширим разнообразието на резултатите, за да получим по-точни извеждания. Не е от съществено значение какъв е вида на класификаторите, тъй като метода за агрегация е универсален.

Нека разгледаме матриците на объркаността за всеки от първоначалните девет модела и за извеждания с агреирация на моделите по метода от предложения метод в Раздел 7, както



Фигура 1: Илюстрация на подхода за извеждане. Тренирани са по три модела от resnet18, resnet34 и resnet50, но това не е съществено за процедурата на агрегация преди извеждането.

и агрегации с максимум и средно. 2, 3, 4, 5, 6 и 7. представляват матриците на объркаността за множество класове за тези извеждания. Това е разширение на концепцията за двоична матрица на объркаността при задачи за класификация с повече от два класа. Вместо да има само две класификации (положителна и отрицателна), в многокласовата матрица на объркаността има редове и стълбове, съответстващи на всеки от класовете в задачата за класификация.

Елементите на матрицата на объркаността за множество класове могат да бъдат описани както следва:

- Истински положителни (True Positives - TP): Брой на случаите, когато моделът е предсказал правилно примери от даден клас като част от този клас.
- Истински отрицателни (True Negatives - TN): Брой на случаите, когато моделът е предсказал правилно отрицателни примери от всички останали класове като част от тях.
- Лъжливи положителни (False Positives - FP): Брой на случаите, когато действителният клас на примера не е предвиден правилно.
- Лъжливи отрицателни (False Negatives - FN): Брой на случаите, когато предсказанието на модела не съвпада с действителния клас.

Така матрицата на объркаността за множество класове е квадратна матрица с размери, равни на броя на класовете в задачата на класификацията. Всяко състояние на класификацията се съдържа в матрицата на объркаността, което позволява детайлно оценяване на производителността на модела за всеки отделен клас.

Резултатите след агрегация по метода описан в Раздел 7, са представени във Фигура 2а. Агрегирането значително подобрява броя правилни предвиждания (главния диагонал на матрицата на объркването), спрямо индивидуалните класификатори, показани по-долу. Наблюдаваме подобрение, което варира от 27 до 103 точни предвиждания повече, спрямо оригиналните девет модела.

8 Практична имплементация на метода за извеждане

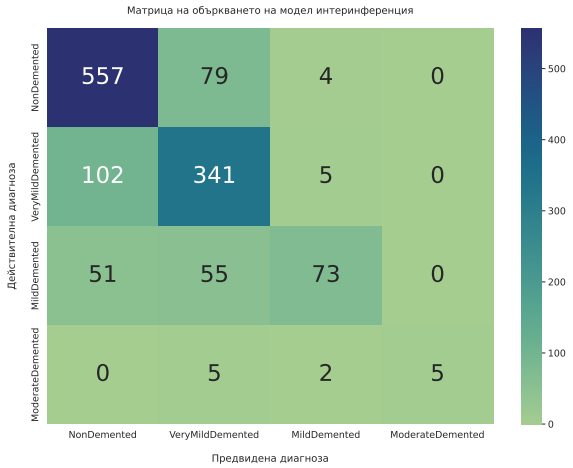
За рангове с малка дължина един начин да се изчисли оптимална агрегация е чрез сравнение на резултатите от всички възможни рангове – подход чрез метода на грубата сила [3].

В Приложение Б функцията

␣␣aggregate_rank_brute

реализира на Python този подход.

Този подход има алгоритмична сложност $o(n!)$ поради нуждата да се обходят всички пермутации на категориите.

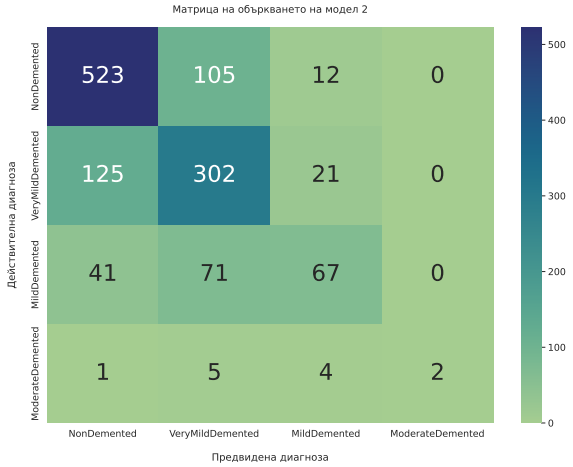


(а) Резултатите след агрегация по метода, описан в Раздел 7. Агрегирането значително подобрява броя правилни предвиждания (главният диагонал на матрицата на объркването), спрямо индивидуалните класификатори, показани по-долу.

(б) *ResNet18* модел има повече грешки спрямо Фигура 2а.

Фигура 2: Резултати на различните методи за извеждане.

(а) *ResNet36* модел има повече грешки спрямо Фигура 2а.



(б) *ResNet50* модел има повече грешки спрямо Фигура 2а.

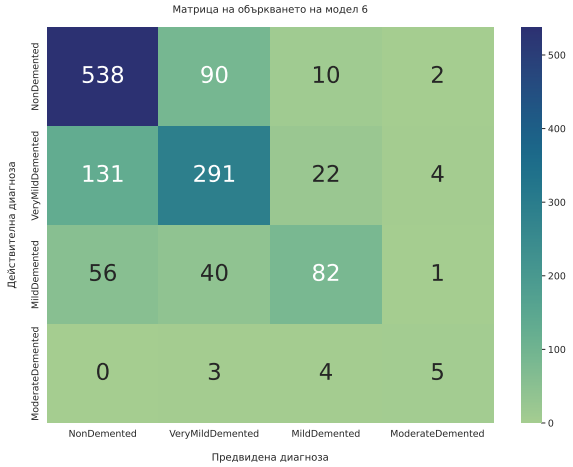
Фигура 3: Резултати на различните методи за извеждане.

(а) *ResNet18* модел има повече грешки спрямо Фигура 2а.

(б) *ResNet36* модел има повече грешки спрямо Фигура 2а.

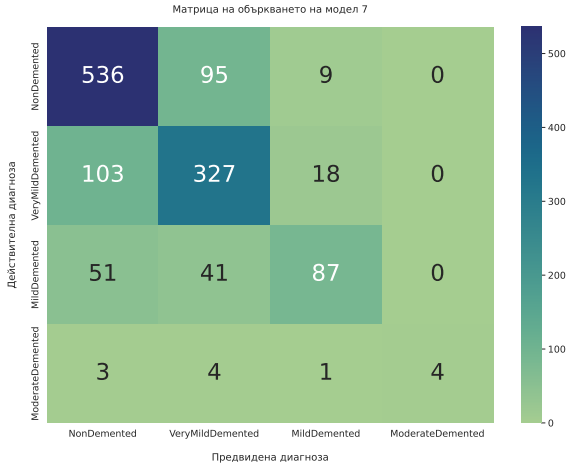
Фигура 4: Резултати на различните методи за извеждане.

(а) *ResNet50* модел има повече грешки спрямо Фигура 2а.



(б) *ResNet18* модел има повече грешки спрямо Фигура 2а.

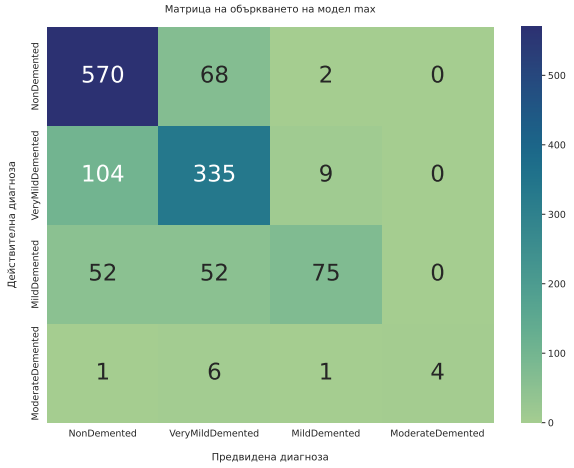
Фигура 5: Резултати на различните методи за извеждане.



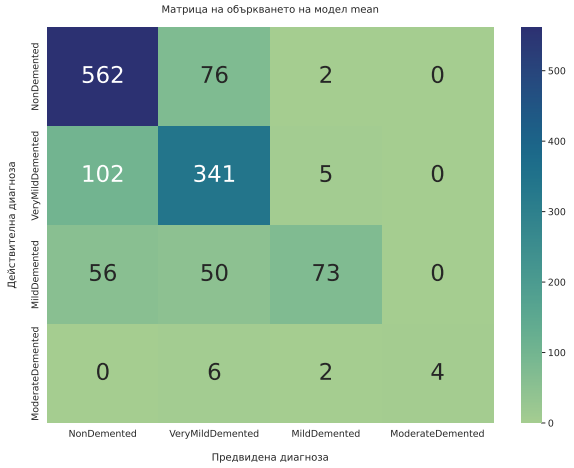
(а) *ResNet36* модел има повече грешки спрямо Фигура 2а.

(б) *ResNet50* модел има повече грешки спрямо Фигура 2а.

Фигура 6: Резултати на различните методи за извеждане.



(а) Агрегиран извеждане с взимане на примера с максимален резултат. Резултатите са сравними с Фигура 2а.



(б) Агрегиран извеждане с взимане на средния резултат. Резултатите са сравними с Фигура 2а.

Фигура 7: Резултати от други методи за извеждане.

8.1 Формулиране на задача за целочислено оптимизиране

Подходът чрез метода на грубата сила е лесен за разбиране и бърз за програмиране, но този подход бързо става нереализуем за проблеми с повече от 10 категории. За съжаление този проблем (заедно с множество свързани проблеми в областта на ранговете) е NP-труден дори само за четири избиратели [7]. Съществуват множество подходи и приближения, които да го направят практичен. В много случаи, като агрегацията на резултатите от търсене, приближенията са достатъчно добри. За случаите, когато се изисква точно решение, формулирането на целочислена оптимизационна задача с ограничения е дадено в [3].

Нека разгледаме точния подход, споменат по-горе. Построяваме насочен граф с тегла $G = (V, E)$ с категориите като върхове. Ребрата се дефинират по следния начин: за всяка двойка кандидати i, j, нека #>{i>j}, означава броя на избирателите, които класират i по-високо от j. Поставяме ребро между всяка двойка i, j с тегло $w_e = |\#\{i > j\} - \#\{j > i\}|$ (ако не е нулево). Ориентацията на реброто е от по-малко предпочитан към по-предпочитан връх.

Формулирането се базира на алтернативното тълкуване на оптималната агрегация по Кемени–Йънг, което минимизира теглата на ребрата, с които не се съгласява:

$$\min \sum_{e \in E} w_e x_e$$

при условието:

$$\forall i \neq j \in V, x_{ij} + x_{ji} = 1 \forall i \neq j \neq k \neq i \in V, x_{ij} + x_{jk} + x_{ki} \geq 1$$

В горепосочената задача всички променливи са цели двоични числа, Те имат следното тълкуване:

$x_{ij} = 1$, ако в агрегирания ранг i е класиран по-ниско от j.

Ограниченията съществено налагат, че променливите дефинират тотална наредба. Първият набор от ограничения налага антисиметрия и пълнота: или i е класиран по-ниско от j, или обратното. Вторият набор от ограничения налага транзитивност [3].

Програмният код може да бъде намерен в Приложение refapp:B с функцията

␣␣aggregate_rank_lp

Използвана е библиотеката ortools [11] от компанията Google.

9 Заключение

Предложеният метод за интекритериална извеждане дава консистентни резултати, които по-начина на построение са устойчиви на грешки. Предложен е и метод за прилагане на прагови стойонсти към принаделжността и неопределеността, който значително повишава точността на селектираните резултати.

A Код за интеркритериални оценки на ансамбъл от класификатори

```
 def count_disagreements(k, l, j):
 n = len(l)
 assert len(k) == n
 return sum(
    k[i] <= k[j] and l[i] > l[j] or k[i] > k[j] and l[i] <= l[j]
    for i in range(n) if i != j
 )


def count_equals(k, l, j):
 n = len(l)
 assert len(k) == n
 return sum(k[i] == k[j] and l[i] == l[j] for i in range(n) if i != j)


def count_agreements(k, l, j):
 n = len(l)
 assert len(k) == n
 return sum(
    not (k[i] <= k[j] and l[i] > l[j] or k[i] > k[j] and l[i] <= l[j])
    and not (k[i] == k[j] and l[i] == l[j])
    for i in range(n) if i != j
 )


from dataclasses import dataclass


@dataclass
class IntervalNumber:
   number: float
   epsilon: float = 0.25
   def __lt__(self, other):
      return self.epsilon < self.number < other.number

   def __le__(self, other):
      return self.epsilon + self.number <= other.number
```

```
    def __gt__(self, other):
        return self.number > other.number + self.epsilon

    def __ge__(self, other):
        return self.number >= other.number + self.epsilon

    def __eq__(self, other):
        return abs(self.number - other.number) <= self.epsilon

    def __ne__(self, other):
        return abs(self.number - other.number) > self.epsilon


assert IntervalNumber(0.5) == IntervalNumber(0.65)
assert IntervalNumber(0.5) < IntervalNumber(0.9)
assert IntervalNumber(0.5) <= IntervalNumber(0.75)
assert IntervalNumber(0.5) != IntervalNumber(0.76)
assert IntervalNumber(0.95) >= IntervalNumber(0.7)
assert IntervalNumber(0.88) > IntervalNumber(0.5)


def inter_score(ranks, y_pred):
    disagreements = 0
    equals = 0
    agreements = 0
    for k, l in combinations(range(n_models), 2):

        k_int = [IntervalNumber(v.item()) for v in ranks[k, :]]
        l_int = [IntervalNumber(v.item()) for v in ranks[l, :]]
        disagreements += count_disagreements(k_int, l_int, y_pred)
        equals += count_equals(k_int, l_int, y_pred)
        agreements += count_agreements(k_int, l_int, y_pred)
    total = agreements + disagreements + equals
    return agreements/total, disagreements/total
```

Б Код за ефективно пресмятане на новия метод за извеждане с ortools

```
from itertools import combinations, permutations

import numpy as np
from ortools.sat.python import cp_model
```

```
def build_graph_from_ranks(ranks):
    n_candidates = ranks.shape[-1]
    edge_weights = np.zeros((n_candidates, n_candidates))
    for rank in ranks:
        for i, j in combinations(range(n_candidates), 2):
            if rank[i] > rank[j]:
                edge_weights[j, i] += 1
            elif rank[i] < rank[j]:
                edge_weights[i, j] += 1
    return edge_weights

def _build_graph_pairwise(pairwise_preferences, n_candidates):
    edge_weights = np.zeros((n_candidates, n_candidates))
    k = 0
    for i, j in combinations(range(n_candidates), 2):
        preference = pairwise_preferences[k]
        if preference > 0:
            edge_weights[j, i] = 1
        else:
            edge_weights[i, j] = 1
        k += 1
    return edge_weights

def aggregate_rank_lp(edge_weights):
    """Kemeny-Young optimal rank aggregation"""
    n_candidates = len(edge_weights)
    # minimize sum_{i,j} w_i_j * x_i_j
    # w_i_j the weight of the edge
    model = cp_model.CpModel()
    x = {}
    for i in range(n_candidates):
        for j in range(n_candidates):
            x[(i, j)] = model.NewIntVar(0, 1, f'x_{i}_{j}')

    for i, j in combinations(range(n_candidates), 2):
        model.AddExactlyOne(x[i, j], x[j, i])

    # and for every cycle of length 3 to ensure transitivity
    for i, j, k in permutations(range(n_candidates), 3):
        model.AddAtLeastOne(x[i, j], x[j, k], x[k, i])

    model.Minimize(
        sum(
```

```
            edge_weights[i, j] * x[i, j]
            for i in range(n_candidates)
            for j in range(n_candidates)
        )
    )
    solver = cp_model.CpSolver()
    solver.parameters.linearization_level = 0
    # Enumerate all solutions.
    solver.parameters.enumerate_all_solutions = True
    status = solver.Solve(model)
    if status == cp_model.OPTIMAL or status == cp_model.FEASIBLE:
#         print(f"Minimum of objective function: {solver.ObjectiveValue()}\n")
        optimal_score = {}
        for i in range(n_candidates):
            optimal_score[i] = sum(solver.Value(x[i, j]) for j in range(n_candidates))
        return solver.ObjectiveValue(), tuple(
            sorted(range(n_candidates), key=lambda i: optimal_score[i])
        )
    else:
        raise ValueError("No solution found.")


def aggregate_rank_brute(predictions, n_classes):
    min_dist = np.inf
    best_rank = None

    for candidate_rank in permutations(range(n_classes)):
        _agreements, disagreements = count_agreements_with_pairs(
            predictions, candidate_rank, n_classes
        )
        dist = disagreements
        if dist < min_dist:
            min_dist = dist
            best_rank = candidate_rank
    return min_dist, best_rank
```

Литература

[1] R. E. Barlow и H. D. Brunk. “The Isotonic Regression Problem and Its Dual”. В: *Journal of the American Statistical Association* 67.337 (1972), с. 140—147.

[2] G. W. Brier. “Verification of Forecasts Expressed in Terms of Probability”. В: *Monthly Weather Review* 78.1 (1950), с. 1—3.

[3] V Conitzer, A Davenport и J Kalagnanam. “Improved Bounds for Computing Kemeny Rankings”. В: 2006.

[4] C. Hallenbeck. “Forecasting Precipitation in Percentages of Probability”. В: *Monthly Weather Review* 48.11 (1920), с. 645—647.

[5] Trevor Hastie, Robert Tibshirani и Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[6] Chih-Wei Hsu и Chih-Jen Lin. “A comparison of methods for multiclass support vector machines”. В: *IEEE Transactions on Neural Networks* 13.2 (2002), с. 415—425.

[7] JJ Bartholdi III, CA Tovey и MA Trick. “The Computational Difficulty of Manipulating an Election”. В: (1989).

[8] John Kemeny. *Mathematics without Numbers*. 1959.

[9] Alexander Marazov. “ОБРАЗНА ДИАГНОСТИКА НА БОЛЕСТТА НА АЛЦХАЙМЕР С КОНВОЛЮЦИОННИ НЕВРОННИ МРЕЖИ: ИМПЛЕМЕНТАЦИЯ С *FASTAI*”. В: *Annual of “Informatics” Section Union of Scientists in Bulgaria* 11 (2021/2022), с. 1—13. ISSN: 1313-6852.

[10] Alexandru Niculescu-Mizil и Rich Caruana. “Predicting good probabilities with supervised learning”. В: *Proceedings of the 22nd international conference on Machine learning* (2005), с. 625—632.

[11] Laurent Perron, Frédéric Didier и Steven Gay. “The CP-SAT-LP Solver”. В: *29th International Conference on Principles and Practice of Constraint Programming (CP 2023)*. Под ред. на Roland H. C. Yap. T. 280. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 3:1—3:2. ISBN: 978-3-95977-300-3. DOI: 10.4230/LIPIcs.CP.2023.3. URL: https://drops.dagstuhl.de/opus/volltexte/2023/19040.

[12] Ryan Rifkin и Aldebaro Klautau. “In Defense of One-Vs-All Classification”. В: *Journal of Machine Learning Research* 5 (2004), с. 101—141.

[13] Tim Roughgarden. *Algorithms Illuminated (Part 4): Algorithms for NP-Hard Problems*. English. Soundlikeyourself Publishing, LLC, юли 2020, с. 273. ISBN: 0999282964.

[14] Telmo Silva Filho и др. “Classifier Calibration: A Survey on How to Assess and Improve Predicted Class Probabilities”. В: *Machine Learning* 112.9 (септ. 2023), с. 3211—3260. ISSN: 1573-0565. DOI: 10.1007/s10994-023-06336-7. URL: https://doi.org/10.1007/s10994-023-06336-7.

[15] Wikipedia. *Kemeny–Young method*. 2024. URL: https://en.wikipedia.org/wiki/Kemeny%E2%80%93Young_method (дата на посещ. 15.03.2024).

[16] H. Peyton Young и Arnold Levenglick. “A Consistent Extension of Condorcet’s Election Principle”. В: *SIAM Journal on Applied Mathematics* 35.2 (1978), с. 285—300.

[17] Peyton Young. “Condorcet’s Theory of Voting”. В: (1988).

[18] Bianca Zadrozny и Charles Elkan. “Transforming classifier scores into accurate multiclass probability estimates”. В: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (2002), с. 694—699.