Instytut Podstawowych Problemów Techniki
Polskiej Akademii Nauk

Rozprawa doktorska

Uczenie aproksymacji funkcji poprzez hierarchiczny podział na podproblemy i łączenie ich rozwiązań

mgr Stanisław Brodowski
Promotor: dr hab. Andrzej Bielecki, prof. AGH

Kraków 2016

Dziękuję wszystkim, których pomoc przyczyniła się do powstania tego doktoratu, w szczególności promotorowi, doktorowi habilitowanemu profesorowi AGH Andrzejowi Bieleckiemu oraz opiekunowi naukowemu, profesorowi Markowi Skomorowskiemu. Dziekuję również Interdyscyplinarnemu Centrum Modelowania Komputerowego Uniwersystetu Warszawskiego za udostępnienie historycznych prognoz pogody.

Streszczenie

Metody uczenia maszynowego służące do predykcji wartości funkcji o przeciwdziedzinie rzeczywistej (aproksymacja funkcji, regresja) mają wiele zastosowań, np. w przewidywaniu zużycia energii elektrycznej, wycenie rozmaitych dóbr czy przewidywaniu rezultatów złożonych procesów, których modele nie są wystarczająco dokładne.

W niniejszej dysertacji zaproponowano algorytm uczenia maszynowego polegający na hierarchicznym podziale całego problemu regresji na częściowo nakładające się podproblemy, utworzeniu względnie prostego rozwiązania dla każdego z nich, a następnie łączeniu odpowiedzi rozwiązań składowych w celu uzyskania odpowiedzi o błędzie mniejszym niż błędy odpowiedzi poszczególnych rozwiązań. Wspomniane podproblemy, a wraz z nimi dotyczące ich rozwiązania, tworzą potencjalnie wielopoziomową strukturę drzewa. Drzewo to tworzone jest automatycznie i w sposób hierarchiczny: jedne składowe rozwiązania powstają w oparciu o inne składowe, wcześniej powstałe, wykorzystując m.in. ich odpowiedzi i błędy. Proponowany w niniejszej pracy algorytm opisany jest w dwóch krokach: schemat algorytmu, który może posiadać różne uszczegółowienia oraz przykład konkretyzacji tego schematu.

W rozprawie udowodnione zostały twierdzenia dotyczące błędu proponowanego w niej rozwiązania, uzasadniające jego poprawność oraz stanowiące wskazówki dla tworzenia konkretyzacji wspomnianego schematu algorytmu.

Proponowana metoda została zastosowana do rozwiązania 21 zadań ogólnie dostępnych, a jej wyniki porównano z wynikami sieci neuronowej (o częściowo dobieranej architekturze) prostego komitetu, metody *bootstrap aggregating* i wzmacniania gradientowego. Rezultaty sugerują ogólny poziom błędu proponowanego rozwiązania zbliżony do najlepszego z rozwiązań porównywanych oraz do najlepszych rozwiązań tej samej kategorii znanych w literaturze, przy czym w niektórych przypadkach błąd proponowanego rozwiązania jest mniejszy. Dla wszystkich tych zadań wartości parametrów kontrolnych proponowanego rozwiązania były takie same. Proponowane rozwiązanie zostało także użyte jako podstawa systemu przewidującego 24-godzinny przebieg zużycia energii elektrycznej na obszarze Polski. Również w tym zadaniu błąd był zbliżony do najlepszych rozwiązań znalezionych w literaturze.

W związku z powyższym uzasadniona jest teza rozprawy, mówiąca, że proponowane w niej rozwiązanie uzyskuje błąd niższy niż pojedynczy składowy element, daje dobre rezultaty, w szczególności nadaje się na podstawę systemu eksperckiego przewidującego przebieg zużycia energii elektrycznej na najbliższe 24 godziny oraz, w wielu zadaniach, nie wymaga dopasowywania parametrów kontrolnych dla osiągnięcia dobrych rezultatów.

Summary

Machine learning methods for estimation of continuous-valued functions (function approximation, regression) are used in many tasks, e.g. prediction of electric power load, appraising of various goods or prediction of results of complex processes that have no accurate enough models.

In this dissertation a general algorithm schema is proposed, that describes the general framework of method of solving continuous-valued function approximation problem, as well as algorithms that concretise that schema. The schema itself is based on combining many simple solutions into one, potentially more accurate. Unit solutions solve overlapping parts of the whole problem. Those parts, as well as solutions, are organized into a, possibly multi-level, tree. That tree is constructed automatically and in a hierarchical manner: making of some of the constituent is based made based on previously constructed parts, using (among other things) their predictions and errors.

In this work theorems are proven concerning errors of the method that validate the solution and allow to form suggestions for the concretisations of the scheme.

The proposed method was applied to 21 tasks found in literature, and its accuracy compared with neural networks, simple committee, bootstrap aggregating and gradient boosting. Results suggest that the general level of error of the proposed method is similar to the best of the compared methods and to the best found in literature, furthermore is better for at least some problems. None of the control parameters of the proposed method was adjusted for particular tasks in the set. The method also formed the basis for system that predicts electrical power load in state power grid 24 hours ahead. In this case, too, results are similar to those of the best solutions found in literature.

Because of aforementioned results, the thesis of this work stating that: the proposed method achieves lower errors than it constituent elements and overall good results, in particular it is suitable to become a basis of a system predicting power load in Polish electric grid for 24 hours ahead, while having little need for adjusting control parameters; is justified.

Spis treści

1. Wstęp

Systemy uczenia maszynowego można wykorzystać do wielu zadań, spośród których wyróżnić można przynajmniej trzy podstawowe rodzaje. W wydobywaniu wiedzy na podstawie przykładów tworzona jest struktura interpretowalna przez człowieka. W przewidywaniu i ocenianiu chcemy poznać prawdopodobną przyszłość lub odzyskać, na podstawie innych, brakujące dane z przeszłości lub teraźniejszości. W przypadku sterowania potrzeba, by program wpływał w pożądany sposób na rzeczywistość. Często zadania ostatniego typu wymagają predykcji i oceny.

Predykcja i ocena, które łączone są tu w jedną grupę (nazywaną dalej predykcją), gdyż ich rozwiązania nieraz mają wiele wspólnego, mogą dotyczyć wielu rodzajów obiektów, jednak najbardziej popularne rodzaje to klasyfikacja (w której odpowiedzią jest przynależność do kategorii) i aproksymacja funkcji, regresja, w której odpowiedziami są liczby rzeczywiste, ew. wektory.

Przykładami zastosowań regresji mogą być: przewidywanie zużycia energii elektrycznej w różnych horyzontach czasowych (od godzin, dni [14, 61], także rozdz. 6, do miesięcy i lat [11]), a także zadania uwzględnione w rozdziale 5, takie jak wycena rozmaitych przedmiotów (domy, samochody), przewidywanie różnych zmiennych dotyczących pracy systemów komputerowych, szacowanie potencjalnych wyników kosztownych badań na podstawie innych, mniej kosztownych, określanie właściwego położenia płaszczyzn sterujących samolotów. Metody uczenia maszynowego w predykcji zmiennej rzeczywistej mogą być też używane do przewidywania zmiennych ekonomicznych, takich jak kursy akcji lub ceny rozmaitych dóbr [79, 102, 103], przewidywania rezultatów skomplikowanych procesów, których modele nie są wystarczająco dokładne [29, 51], czy też przeżywalności pacjentów na podstawie danych medycznych [73, 99].

Bogactwo zastosowań regresji pociąga za sobą wielość rozwiązań. Jednymi z częściej stosowanych są metody statystyczne, sieci neuronowe różnych rodzajów, drzewa modelowe (wspomniane dalej w podrozdziale 1.4), metody minimalnoodległościowe [44]. Każda z nich charakteryzuje się odmiennymi właściwościami i nie ma metody, która zawsze mogłaby być najlepsza [100]. Dlatego nowe metody uczenia maszynowego dla regresji wciąż są pożądane, o ile tylko nie dają gorszych rezultatów dla

wszystkich (lub prawie wszystkich) zadań od metod już istniejących. Dodatkowo wiele z istniejących rozwiązań ma parametry kontrolne znacząco wpływające na ich działanie, a które wymagają dostosowywania pomiędzy zadaniami, przerzucając część odpowiedzialności za naukę na środowisko zewnętrzne (np. użytkownika lub inne algorytmy, w tym wyczerpującego przeszukiwania).

Zastrzeżenia te dotyczą także grupy metod, których przedstawiciel jest opisywany w niniejszej pracy. Są to metody łączące wiele prostszych rozwiązań w jedno, które ma być skuteczniejsze dla danego zadania. W wielu pracach (wymienionych np. w podrozdziale 1.4) pokazano, że istotnie, takie podejście może dawać bardzo dobre rezultaty w postaci dokładniejszych przewidywań lub ocen.

W niniejszej pracy proponowana jest metoda uczenia maszynowego dla predykcji zmiennej rzeczywistej (wektorowej) łącząca wiele prostych i względnie niedokładnych rozwiązań tego zagadnienia w jedno dokładniejsze. Metoda ta, nazywana skrótowo "hierarchiczną aproksymacją", wykorzystuje hierarchiczny podział problemu na nakładające się podproblemy oraz dynamiczne (tzn. zależne od konkretnych danych wejściowych) łączenie rezultatów poszczególnych rozwiązań składowych. Udowadniane są twierdzenia oraz opisywane przeprowadzone eksperymenty dotyczące tej metody.

1.1. Cele i teza pracy

Jak wspomniano we wstępie, łączenie wielu rozwiązań w jedno dające mniejszy błąd może być sposobem na uzyskanie dobrych predykcji, a nowe rozwiązania w tej kwestii są wciąż pożądane, o ile dają dobre rezultaty dla nietrywialnej grupy zadań. Dlatego celem niniejszej pracy było opracowanie różnej od istniejących i dającej dobre rezultaty metody uczenia maszynowego łączącej wiele prostszych rozwiązań dotyczących predykcji zmiennej rzeczywistej (wektorowej) w jedno skuteczniejsze, wykorzystującej hierarchiczny podział problemu na nakładające się podproblemy. Ponadto, metoda ta powinna wymagać niewiele wkładu użytkownika w postaci dopasowywania parametrów kontrolnych. Metoda wykazuje cechy podobne do niektórych rozwiązań dostępnych w literaturze (zob. podrozdział 1.4), ale sposób ich połączenia jest oryginalny.

Teza niniejszej rozprawy brzmi następująco:

Proponowana metoda, będąca połączeniem metod "dziel i zwyciężaj" oraz komitetów:

1. Uzyskuje niższy błąd niż pojedynczy składowy element.

2. **Daje dobre rezultaty, tzn. porównywalne ze znanymi technikami łączenia klasyfikatorów w regresji. W szczególności nadaje się na podstawę systemu ekspertowego przewidującego profil zużycia energii elektrycznej na najbliższe 24 godziny.**
3. **Wymaga dopasowywania parametrów kontrolnych jedynie w niewielkim zakresie lub nie wymaga go wcale dla osiągnięcia dobrych rezultatów w wielu zadaniach.**

1.2. Zakres i zawartość pracy

W dalszej części tego rozdziału można znaleźć wstępne określenie cech proponowanego w tej dysertacji rozwiązania oraz ich porównanie z cechami metod dostępnych w literaturze. W następnym rozdziale wprowadzone zostaną bardziej formalnie podstawowe pojęcia oraz przedstawiony ogólny schemat algorytmu, opisane uprzednio w pracy autora tej rozprawy [24] (tu z drobnymi modyfikacjami). W rozdziale 3 przytoczono oryginalne twierdzenia dotyczące skuteczności proponowanego rozwiązania, dowodzące prawdziwości pierwszej części tezy pracy. Niektóre z nich pochodzą z wcześniejszych prac autora niniejszej rozprawy [21, 24]. W kolejnym rozdziale (4) zaproponowano nowe metody konkretyzujące ogólny schemat i będące modyfikacjami rozwiązań proponowanych w artykułach autora niniejszej rozprawy [23, 24]. Na końcu rozdziału 4 oszacowano złożoność czasową algorytmu. W rozdziale 5 opisano przeprowadzone dla tej pracy eksperymenty na 21 ogólnie dostępnych [8] zadaniach uczenia maszynowego, natomiast w rozdziale 6 sprawdzono użyteczność rozwiązania dla zadania przewidywania profilu zużycia energii elektrycznej w Krajowym Systemie Elektroenergetycznym. Ostatni rozdział (7) zawiera podsumowanie pracy, w tym możliwe kierunki rozwoju proponowanego rozwiązania. Po nim następują dodatki, w których zamieszczono dowody twierdzeń (dod. A) oraz szczegółowe tabele z wynikami eksperymentów (dod. B).

1.3. Cechy proponowanego rozwiązania

Opisane w niniejszej pracy rozwiązanie dzieli się na dwa poziomy – schemat algorytmu określający ogólną strukturę rozwiązania i konkretne algorytmy wypełniające tę strukturę. Rozwiązanie posiada kilka charakterystycznych cech wartych wspomnienia na początku.

Zgodnie z celami pracy, głównym motywem schematu algorytmu jest łączenie wielu względnie prostych (takich jak sieci neuronowe z kilkoma neuronami ukrytymi) rozwiązań podproblemów w rozwiązanie całego problemu uzyskujące błąd mniejszy niż poszczególne składowe.

Tworzenie rozwiązania odbywa się w sposób hierarchiczny: jedne składowe rozwiązania powstają w oparciu o inne, wcześniej powstałe składowe. Jednak, w odróżnieniu od wielu rozwiązań wymienionych w podrozdziale 1.4, zależność ta polega tylko na podziale zbioru uczącego oraz przydziale przykładów do późniejszej oceny, nie zaś np. na zmianie funkcji celu. Oznacza to podział głównego problemu na pewne podproblemy. Jest to wyraz strategii "dziel i zwyciężaj", szeroko znanej w informatyce.

Wspomniany podział na podproblemy dokonuje się automatycznie poprzez dopasowanie ich zakresów do zadania. Podproblemy te ponadto nakładają się, tak by przykłady były poddawane ocenie wielu bazowych rozwiązań, co zmniejsza błąd, w sposób znany z komitetów (zob. podrozdział 1.4). Dlatego można powiedzieć, iż proponowane rozwiązanie, oprócz cech algorytmów "dziel i zwyciężaj", posiada też cechy komitetów. Podproblemy dzielone są rekursywnie do osiągnięcia warunku stopu i tworzą strukturę drzewa. Skutkiem tego strukturę drzewa tworzą również rozwiązania składowe. Każdy węzeł w drzewie rozwiązuje pewien podproblem danego problemu, nie tylko liście (jak w niektórych rozwiązaniach wspomnianych w podrozdziale 1.4).

W czasie uzyskiwania wyników rozwiązania wynik całości powstaje poprzez ważone uśrednianie wyników poszczególnych komponentów. Wagi te obliczane są dynamicznie dla każdego przykładu, przy czym mogą być tu użyte zmienne wejściowe, wyjściowe, a także wyniki rozwiązania składowego umieszczonego w rodzicu węzła.

1.4. Niniejsza praca w kontekście innych rozwiązań z tego obszaru

Tematyka łączenia wielu prostszych rozwiązań uczenia maszynowego w jedno jest obecna w literaturze co najmniej od początku lat dziewięćdziesiątych ubiegłego wieku [38, 89], niektóre teoretyczne podstawy powstawały nawet wcześniej [30]. Duża część z tych prac jest związana przede wszystkim z klasyfikacją, jak np. [76, 77, 86, 89]. Dla niektórych metod udowodniono teoretycznie (za: [25, 67]) i wykazano empirycznie (np. [12, 32, 46, 51]), że takie podejście może dawać dobre rezultaty. Ponieważ klasyfikacja w tym kontekście ma swoją, różną od regresji spe-

cyfikę, przywołane tu zostaną przede wszystkim rozwiązania przydatne dla regresji, z wyjątkiem Hierarchicznego Klasyfikatora [76].

1.4.1. Metody uśredniające

Jednym z najprostszych rozwiązań łączących wiele modeli regresji w jeden jest prosty jednopoziomowy komitet [67, 93]. Składa się on z pewnej liczby rozwiązań bazowych, których wyniki są łączone. Najprostszym z kolei komitetem jest po prostu zestaw ustalonej liczby rozwiązań podstawowych, których wyniki są uśredniane. Nawet taka kombinacja może dawać lepsze wyniki niż pojedyncze rozwiązanie [46], o ile błędy składowych się różnią [93] (mają tu zastosowanie także niektóre twierdzenia z rozdz. 3, np. tw. 1), co jest dość łatwo spełnione w przypadku rozwiązań mających składowe stochastyczne (jak np. inicjalizacja wag sieci neuronowych). Ogólnie znane są natomiast metody ustalania wag dające potencjalnie lepsze rezultaty [42, 46, 67], w szczególności niekiedy lepiej jest pominąć sieci, które nie pomniejszają błędu komitetu [104]. W bardziej zaawansowanych rozwiązaniach [51, 64] kwestią staje się także tworzenie poszczególnych rozwiązań składowych. Zakładając, że w zwykłym komitecie nie manipulujemy danymi (to domena rozwiązań opisanych poniżej), można wciąż powiększyć różnice pomiędzy rozwiązaniami, potencjalnie zmniejszając błąd całego komitetu. Przykładem jest tu tzw. negatywna korelacja [64], stosowana przede wszystkim dla sieci neuronowych, wprowadzająca do funkcji błędu karę za posiadanie błędów skorelowanych z innymi sieciami. Znane są też zaawansowane algorytmy konstrukcyjne dla komitetów, np. [51].

Komitety tego typu, podobnie jak proponowane rozwiązanie, wykorzystują uśrednianie wielu rozwiązań w celu polepszenia wyniku, ale nie tworzą struktury drzewa ani nie starają się skonstruować łatwiejszych zadań dla poszczególnych składowych przez wydzielenie im części przestrzeni, na której mogą się specjalizować.

1.4.2. Wzmacnianie i wzmacnianie gradientowe

Kolejną grupę bardziej zaawansowanych metod łączenia stanowią te oparte na tzw. "wzmacnianiu" (boosting) z najbardziej znanym algorytmem *AdaBoost* przeznaczonym do klasyfikacji [37, 66, 89, 93, 101]. Bazuje on na dodawaniu kolejnych podstawowych rozwiązań, przy uczeniu których większą wagę mają przykłady błędnie klasyfikowane w dotychczasowym zestawie. Odpowiedź całości dla zadanego przykładu powstaje przez głosowanie poszczególnych składowych, przy czym większa waga przypisywana jest rozwiązaniom o mniejszym błędzie. Bazowe rozwiązania są "słabe", co w tym przypadku oznacza, że ich błąd może być dowolnie mniejszy

od błędu spowodowanego losowym przypisaniem klas [37].

Od tego czasu powstało wiele wersji dotyczących regresji [12, 32, 37, 48, 85]. W algorytmie *AdaBoost.R* zaproponowanym w [37] *AdaBoost* jest wykorzystywany poprzez sprowadzenie regresji do klasyfikacji binarnej w rodzaju "czy wynik jest większy niż x". Poprzez manipulację wagami uzyskuje się bezpośrednią zależność pomiędzy błędem średniokwadratowym a wspomnianą klasyfikacją binarną. Algorytm ten był modyfikowany, przykładem może być *AdaBoost.R2* [32] ze zmienionym systemem ważenia oraz uśredniania. Algorytmy te są często używane z drzewami (M5 [80] lub decyzyjnymi) jako estymatorami bazowymi, ale nie jest to koniecznością.

Dostępne są również inne rozwiązania tej klasy, np. [12] łączący regresje za pomocą mediany i używający progowania, aby zmniejszać liczność błędów większych od odpowiednio dobranego progu, czy algorytm AdaBoost.RT [90] wykorzystujący dzielenie przypadków na klasy "dobrze" i "źle" ocenione ze względu na przekroczenie progowego względnego błędu na moduł (inaczej niż większość prac uprzednio cytowanych w tym punkcie) i średnią arytmetyczną do łączenia predykcji poszczególnych modeli.

Metoda gradientowego wzmacniania regresji (*gradient boosting machine*, dostępna też dla klasyfikacji) dla błędu średniokwadratowego [39] przy tworzeniu kolejnych elementów zestawu również skupia się na błędzie elementów wcześniej utworzonych, ale w inny sposób. Tutaj kolejne rozwiązania uczą się negować błąd dotychczas utworzonego zestawu. W celu polepszenia generalizacji, poprawki kolejnych zmiennych mogą być uwzględniane nie całkowicie, ale pomnożone przez parametr regularyzacyjny $0 < \alpha \leqslant 1$. Formuła poprawek jest analogiczna z optymalizacją za pomocą największego spadku gradientu. Powstała również wersja stochastyczna, losująca zbiory uczące kolejnych rozwiązań bazowych (bez zwracania) [40]. Rozwiązanie zaprezentowane w pracy [34] łączy cechy obu wspomnianych grup "wzmacniania", modyfikując zarówno skład zbiorów uczących kolejnych rozwiązań w zależności od błędu poprzednich, jak i funkcję celu.

Rozwiązania z dziedziny *boosting* łączą wiele "słabych", prostych rozwiązań w jedno większe oraz, jak niektóre z wariantów proponowanego w niniejszej rozprawie rozwiązania, uwzględniają błąd już wygenerowanych elementów przy tworzeniu kolejnych. Nie dzielą jednak przestrzeni na podprzestrzenie, nawet jeśli zmieniają zbiory uczące, i nie tworzą struktury drzewa w taki sposób, w jaki czyni to proponowana w niniejszej pracy metoda.

1.4.3. *Bagging*

Różnica pomiędzy dwoma ostatnimi rozwiązaniami z poprzedniego punktu przypomina metodę agregowania rozwiązań uczonych na częściowo losowych zbiorach "Bootstrap aggregating" (*bagging*) [16, 93], stosowaną również w regresji dla metod niestabilnych, czyli takich, które wykazują dużą zmienność wyniku ze względu na zbiór uczący [67]. Polega ona na losowaniu przykładów do zbiorów uczących aproksymacji bazowych ze zwracaniem, zwykle w liczbie równej liczności oryginalnego zbioru (ew. mniejszej). Zaletą tej metody jest możliwość monitorowania błędu generalizacji przy użyciu błędów poszczególnych rozwiązań na przykładach, które nie znalazły się w ich zbiorach uczących ("out-of-the-bag"; ze względu na losowanie ze zwracaniem).

W wersji iterowanej powyższego algorytmu, *Iterated Bagging* [18], rozwiązania składowe w kolejnych etapach uczone są nie pierwotnej funkcji, ale korekt odpowiedzi systemu utworzonego w poprzednich krokach – podobnie do wzmacniania gradientowego. Kolejnymi składowymi rozwiązaniami są tu instancje zwykłego *bagging*u, a korekty są obliczane tylko względem tych podstawowych rozwiązań, których zbiory uczące nie posiadały danego przykładu. Tak skonstruowane rozwiązanie ma większe szanse uzyskiwać błąd na nieznanych przykładach niższy, niż gdyby do obliczania poprawek zostały użyte wszystkie dostępne w danym momencie rozwiązania składowe [18].

Rozwiązania tego rodzaju łączą wiele rozwiązań modyfikując zbiory uczące, ale (podobnie jak "boosting") nie dzielą przestrzeni na podprzestrzenie w taki sposób, jak rozwiązanie proponowane w niniejszej pracy i nie tworzą struktury drzewa.

1.4.4. Drzewa decyzyjne i modelowe

Innym od opisanych powyżej podejściem jest podział przestrzeni wejściowej na fragmenty, na których nawet proste rozwiązania dawałyby wystarczająco mały błąd, a następnie połączenie ich wyników, przede wszystkim przez wybór odpowiedniego z tych prostych rozwiązań. Za analog matematyczny tej klasy metod można uznać funkcje sklejane. Jednymi z pierwszych i najbardziej znanych [101] rozwiązań uczenia maszynowego, które wykorzystywały to podejście, były drzewa klasyfikacji i regresji *Classification and Regression Trees* [19]. Ze względu na wielką prostotę klasyfikatorów w węzłach (tylko w liściach), którymi są po prostu wartości stałe (średnie), w zasadzie nie stosowane w praktyce jako samodzielne podstawy predykcji, przypadek ten można uznać za graniczny pomiędzy klasyfikatorem jednolitym a kombinacją. Natomiast w drzewie M5 [80] węzły reprezentują wielowymiarową regresję liniową – metodę, która bywa stosowana jako samodzielna do rozwiązywania prostych pro-

blemów.

Uczenie tego drzewa polega na iterowanym podziale podprzestrzeni modelowanej przez jeden z obecnych liści na dwie podprzestrzenie, w taki sposób aby regresje liniowe w tak utworzonych węzłach jak najbardziej redukowały błąd rozwiązania. Wspomniane regresje liniowe mogą z zasady używać tylko niektórych zmiennych wejściowych. Takie postępowanie jest kontynuowane, dopóki możliwe redukcje nie staną się mniejsze niż założony parametr. Po utworzeniu wstępnego drzewa może ono zostać zredukowane tak, aby zmniejszyć oszacowanie jego błędu na przykładach nieobecnych w zbiorze uczącym (tzw. *pruning*). Oszacowanie to zależy od liczby przykładów uczących danego węzła oraz liczby parametrów modelu. Dokładne określenie punktu zatrzymania oraz ew. późniejszego usuwania węzłów pozostają wciąż istotnymi kwestiami.

Proponowane w niniejszej pracy rozwiązanie również tworzy strukturę drzewa i również dzieli podprzestrzeń na obszary, na których potencjalnie mogą być zastosowane prostsze rozwiązania. Jednak tutaj istotne jest założenie nakładania się przestrzeni (które nawet jeśli w przytaczanych metodach używane, np. wygładzanie (*smoothing*), jest znacznie mniej istotne). Ogólna postać proponowanego w niniejszej dysertacji rozwiązania jest mniej zależna od stosowanego sposobu podziału i konkretnego rodzaju aproksymatora w węźle. Rozwiązania tych dwóch problemów przedstawione w niniejszej pracy są wysoce nieliniowe i powodują wyraźne różnice w podejściach, dają też niższe błędy na problemach nieliniowych (zob. rozdz. 5).

1.4.5. Lasy drzew

Losowy Las (*random forest*) używa techniki *bagging* na lasach drzew decyzyjnych (podobnie zresztą jak wiele zastosowań zwykłego *bagging*). Dodatkowo, w węzłach każdego z drzew decyzyjnych używany jest losowy i niezależny od innych podobnych zbiór cech, względem których można dzielić przestrzeń wejściową dla poszczególnych rozwiązań bazowych [17, 62].

Las obrotów (*rotation forest*) [58] również używa techniki *bagging* dla drzew, ale zamiast losować podzbiór cech do użycia, losuje kilka zestawów cech, na każdym uruchamia analizę składowych głównych, wybierając następnie najlepsze cechy ze wszystkich zestawów. Na takim losowym zestawie przykładów i losowo obróconym zestawie cech tworzone jest drzewo. Zastosowanie lasu obrotów do regresji opisane jest np. w [73].

Te techniki łączą predykcje wielu drzew, stąd w pewien sposób wykonują dzielenie przestrzeni na części, na których łatwiej będzie się uczyć (o ile mówimy o drze-

wach modeli podobnych do M5 [80]), uśredniając ich odpowiedzi. Mają jednak wyraźnie różną strukturę, inne założenia dotyczące najmniejszych elementów składowych, a więc i sposoby uczenia od proponowanego rozwiązania.

1.4.6. Mieszaniny ekspertów

Podobnie jak rozwiązania z dwu powyższych grup, hierarchiczna mieszanina ekspertów (*Hierarchical Mixture of Experts* [53]), oparta na wcześniejszej, jednopoziomowej strukturze mieszaniny ekspertów [52, 53], również tworzy strukturę drzewa. W liściach owego drzewa są tzw. eksperci – uogólnione regresje liniowe (perceptrony bez warstwy ukrytej, z ew. transformacją nieliniową na wyjściu), w przypadku zadania regresji często po prostu liniowe. W węzłach wewnętrznych (tzw. bramkach) znajdują się natomiast wektory ważące, po jednym dla każdego węzła dziecka. Wagę danego węzła dziecka w danym węźle wewnętrznym stanowi wartość funkcji wykładniczej od iloczynu skalarnego odpowiadającego węzłowi potomnemu wektora z danym wektorem wejściowym, znormalizowana tak, aby sumy wag dla każdego przykładu sumowały się do jeden (tzw. miękkie maksimum). Taki model pozwala na interpretację probabilistyczną [53]. Bramki i eksperci uczeni są łącznie np. metodą gradientową lub wartości oczekiwanej – maksymalizacji (*expectation-maximalization* – EM) [53]. Choć początkowo struktura drzewa była ustalona z góry, powstały algorytmy tworzące ją automatycznie [88, 96].

Hierarchiczne mieszaniny ekspertów (HME) mają wiele wspólnych cech z proponowanym w rozprawie rozwiązaniem, takich jak tworzenie drzewa (w przypadku algorytmu konstruktywnego), nakładanie się obszarów, za które odpowiadają końcowe rozwiązania, czy użycie średniej ważonej ekspertów (zamiast bardziej ogólnej kombinacji liniowej). Istnieją też wersje dokonujące podziału na podproblemy [82], zatem również łączą cechy rozwiązań "dziel i zwyciężaj" i komitetów. Intensywne wykorzystanie paradygmatu probabilistycznego do konstrukcji tych rozwiązań skutkuje jednak istotnie różnymi od opisywanej w niniejszej pracy metody algorytmami uczenia, preferującymi względnie prostych (np. liniowych) ekspertów w liściach. Inna jest też rola węzłów wewnętrznych. W HME są one jedynie bramkami, natomiast w proponowanej w niniejszej pracy metodzie rozwiązują one pewien podproblem ogólnego problemu i, oprócz ważenia odpowiedzi wyników dzieci, decydują o ich zbiorach uczących w czasie uczenia, a w czasie działania ich odpowiedzi dla danego przykładu są składowymi ostatecznej odpowiedzi.

1.4.7. Rozwiązania rozmyto-neuronalne

Z szerokiej klasy rozwiązań hybrydowych neuronalno-rozmytych [28, 31, 95] z punktu widzenia niniejszej rozprawy najbardziej interesujące są prace, w których, tak jak w przypadku [28] czy [31], zbiory rozmyte używane są do podziału przestrzeni na części w sposób ułatwiający uczenie. W pierwszym ze wspomnianych rozwiązań jego pierwszy poziom skonstruowany jest z klasyfikatora dzielącego na zadaną z góry liczbę klas. Klasy wyodrębniane są jako przedziały zmiennej wyjściowej o równej długości ("niska wartość", "wysoka wartość" itp.). Drugi poziom tworzą aproksymacje funkcji (sieci neuronowe lub regresja oparta na wektorach podpierających) starające się odtworzyć kształt funkcji w tych przedziałach. W [31] w liściach również znajdują się sieci neuronowe, ale przestrzeń dzielona jest w sposób rozmyty na 4 części w dwóch wymiarach, a następnie proces może być powtórzony na kolejnych poziomach. Uśrednianie używa jako wag znormalizowanych poziomów przynależności.

Wspomniane systemy, podobnie jak proponowane w niniejszej pracy rozwiązanie, dzielą przestrzeń z intencją ułatwienia zadania rozwiązaniom składowym. Wyniki ważone są normalizowanym stopniem przynależności, jak można interpretować także zachowanie proponowanego rozwiązania. Jednak, tak jak w poprzednio wymienionej klasie rozwiązań, zasadniczo węzły wewnętrzne nie rozwiązują bezpośrednio części problemu, a jedynie rozdzielają przykłady. Ponadto w węzłach tych zastosowanie zbiorów rozmytych jest bardziej bezpośrednie (czasem nieautomatyczne) niż w przypadku proponowanego w rozprawie rozwiązania, w którym używane jest tylko grupowanie rozmyte (podrozdział 4.1).

1.4.8. Hierarchiczny klasyfikator

Rozwiązaniem dotyczącym ściśle klasyfikacji wieloklasowej, o którym wspomnienie tu jest konieczne, jest hierarchiczny klasyfikator [75, 76, 77]. Rozwiązanie to tworzy drzewo nakładających się podproblemów, które rozwiązywane są za pomocą tzw. słabych klasyfikatorów. Podział oparty jest na grupowaniu macierzy pomyłek klasyfikatora rodzica tak, aby klasyfikatory w węzłach miały do rozróżnienia mniej klas, ale aby jednocześnie dla jednego przykładu było wywoływanych kilka klasyfikatorów, co zmniejsza szansę na uzyskanie błędnej odpowiedzi. Było ono inspiracją dla powstania hierarchicznej aproksymacji (jak nazywać można rozwiązanie proponowane w niniejszej pracy), jednak istotne różnice między klasyfikacją wieloklasową i zadaniem przewidywania wartości funkcji skutkują znaczącymi różnicami pomiędzy rozwiązaniami. Obecnie, mimo iż mają wspólnych wiele koncepcyjnych cech, jak podział na nakładające się podprzestrzenie, struktura drzewa czy uśrednianie

wyników, różnią się nie tylko szczegółowymi rozwiązaniami, ale także ogólnym sformułowaniem podstawowych zasad.

1.4.9. Podsumowanie

Jak można wywnioskować nawet z powyższego przeglądu literatury, występuje wiele rozwiązań dotyczących łączenia prostszych zadań regresji. Można je podzielić na dwie duże grupy (w zgodzie z metodologią występującą też w [67]). Rozwiązania z pierwszej grupy (punkty 1.4.4 – 1.4.7), także inne, takie jak wielowymiarowe adaptujące się funkcje sklejane *multivariate adaptive regression splines (MARS)* [38], uwzględniają podział zbiorów uczących (zwykle poprzez podział przestrzeni) na mniejsze podgrupy, które mają ułatwić uczenie. Do drugiej grupy należą rozwiązania wspomniane w punktach 1.4.1 – 1.4.3, polegające przede wszystkim na uśrednianiu, być może z uwzględnieniem zmiany funkcji celu lub próbami heterogenizacji poszczególnych ekspertów za pomocą różnych technik [67], w tym zmiany zbiorów uczących, ale bez typowego podziału na podprzestrzenie z zamiarem ułatwienia uczenia. Każde z wymienionych rozwiązań wykazuje istotne różnice z proponowanym rozwiązaniem, również te najbardziej zbliżone: hierarchiczne mieszaniny ekspertów (zob. punkt1.4.6) i hierarchiczny klasyfikator (zob. punkt 1.4.8).

2. Podstawowe definicje i schematy algorytmów

2.1. Wstępne wiadomości i oznaczenia

W tym rozdziale podstawowe pojęcia dotyczące opisywanego rozwiązania wprowadzone zostaną w sposób formalny. Ogólnym sposobem podejścia do przewidywania wartości zmiennych o wartościach rzeczywistych jest znajdowanie relacji typu funkcyjnego pomiędzy zmiennymi wejściowymi, których wartości znamy, a zmiennymi wyjściowymi, których wartości chcemy przewidzieć.

W dalszych rozważaniach zmienna wejściowa będzie zapisywana jako X. Przez $\mathcal{X}$ będzie oznaczona przestrzeń wejściowa, tzn. zbiór wszystkich potencjalnych wartości tej zmiennej - zbiór możliwych wejść, natomiast wartość wejść – wektor cech – zazwyczaj przez x z odpowiednimi indeksami. Analogiczne pojęcia odnoszące się do wyjścia będą oznaczane jako Y, $\mathcal{Y}$ i y. Przez *przykład* rozumiana będzie para (x, y): jedna instancja wejścia i odpowiadające mu wyjście (niezależnie od tego, czy wyjście jest, z punktu widzenia dyskusji, znane, czy nie). W opisywanym systemie przyjęto założenie, że zmienne są wektorami liczb rzeczywistych, czyli $\mathcal{Y} \subset \mathbb{R}^r$ i $\mathcal{X} \subset \mathbb{R}^p$.

Szukana zależność w ogólnym przypadku mogłaby być rozpatrywana jako *relacja* R, tzn. pewien zbiór par wejść i wyjść, takich że

$$(x, y) \in R \iff \text{dla wejścia } x \text{ może wystąpić wyjście } y \tag{2.1}$$

Takie podejście uwzględnia fakt, że jednemu wejściu nie musi odpowiadać zawsze jedno wyjście, ponieważ może zależeć ono od czynników nie ujętych w wejściu, być z natury losowe, bądź mieć błędy pomiaru (podobnie zresztą jak wejście). Nie uwzględnia jednak tego, że różne wyjścia mogą wystąpić dla jednego wejścia z różnymi prawdopodobieństwami.

Sytuacją idealną byłoby, gdybyśmy byli w stanie poznać lub choćby przybliżyć rozkład warunkowy wyjścia dla danego x. Jednak rzadko dla konkretnego wejścia dostępne jest więcej niż jedno wyjście, a same wejścia nie są zwykle gęsto rozmieszczone

w przestrzeni cech. Dlatego stosowane jest w niniejszej pracy bardziej bezpośrednie podejście, zgodnie z którym *predykcja* oznacza podanie dla danego wektora wejściowego konkretnej odpowiedzi, która będzie jak "najlepsza" lub, w jakimś znaczeniu, najbardziej reprezentatywna.

"Najlepsza" wartość w tym przypadku może oznaczać wartość minimalizującą funkcję ryzyka. W ujęciu probabilistycznym jest to wartość oczekiwana (zwykle przy pewnych założeniach co do błędu ε) funkcji straty $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, określającej jak bardzo niepożądany jest konkretny błąd [48]. Natomiast "reprezentatywne" może oznaczać pewną statystykę - np. wartość oczekiwaną. Przy odpowiednich założeniach o rozkładzie warunkowym te wymagania są równoważne (zob. s. 14).

W niniejszej pracy użyty zostanie zapis nawiązujący do znajdowania wspomnianej "najlepszej" wartości. Polega on na rozłożeniu relacji na dwa składniki - zasadniczą zależność funkcyjną f (zależną od X) oraz odchylenie, błąd ε:

$$Y = f(X) + \varepsilon \tag{2.2}$$

Składnik ε bezpośrednio reprezentuje tutaj fakt niecałkowitej zależności funkcyjnej Y od X.

Obydwa sposoby można traktować w sposób probabilistyczny - z jednej strony zakładając istnienie prawdopodobieństwa wystąpienia każdej z par w relacji oraz rozkładu warunkowego Y pod warunkiem konkretnego x lub, równoważnie, traktując ε dla wspomnianej szukanej jako zmienną losową.

Należy tu zauważyć, że dokładne postaci szukanej funkcji oraz rozkład błędu ε są wzajemnie zależne, a więc założenia dotyczące ε wpływają na postać szukanej funkcji i mogą być dobierane w zależności od celu i zastosowań. Przybliżenie funkcji f, które będziemy znajdować, oznaczane będzie jako $\hat{f}$.

Jak już zostało wspomniane, z powodu nieznajomości prawdziwych rozkładów warunkowych nie da się bezpośrednio obliczyć funkcji ryzyka. Dlatego w tej pracy do oceny jakości rozwiązania stosowana jest w różnych miejscach średnia wartość funkcji straty na pewnym ciągu przykładów.

W dalszej części pracy ciąg przykładów, o ile nie będzie należał do bardziej specyficznych kategorii wspomnianych poniżej, będzie oznaczany zazwyczaj jako $\mathbf{S}$ (oczywiście z odpowiednią indeksacją), a jego liczność jako $|\mathbf{S}|$.

Funkcję ryzyka na przykładach spoza zbioru uczącego można oszacować poprzez testowanie, czyli obliczenie funkcji strat na pewnej próbce rzeczywistych danych. Próbka taka, czyli ciąg (lub zamiennie zbiór, jeżeli kolejność nie ma znaczenia) testowy, składająca się z przykładów - par (x, y) - będzie oznaczana jako $\mathbf{T}$.

Ponieważ prezentowana w tej pracy metoda należy do działu uczenia maszynowego, wspomniana wyżej relacja będzie znajdowana na podstawie istniejącego zbioru przykładów o znanych poprawnych wyjściach: zbioru uczącego. Zbiór ten będzie oznaczany przez **U**.

Samo ustalanie rozwiązania w większości przypadków polega na znalezieniu jak najlepiej dopasowanego przedstawiciela z pewnej założonej rodziny funkcji. Rodzina ta może być dana implicite albo explicite, a dopasowania dokonuje się na zbiorze uczącym [48]. W tej pracy model ten jest skutkiem przyjętej architektury rozwiązania i nie jest przedstawiony wzorem.

Używane funkcje strat

Dobór funkcji strat wynika ściśle z zastosowania i względów praktycznych. Istnieje jednak kilka funkcji strat uznawanych za standardowe. W tym rozdziale użyto przede wszystkim średniego błędu kwadratowego, oznaczanego tu jako MSE (od ang. *mean squared error*). Dla zbioru przykładów $\mathbf{S}$ o liczności oznaczonej $|\mathbf{S}|$, gdzie x_i, y_i i $\hat{f}(x_i)$ są odpowiednio i-tym wektorem wejściowym, poprawną odpowiedzią dla i-tego przykładu oraz otrzymanym przybliżeniem funkcji, a $||\cdot||$ normą euklidesową:

$$\mathrm{MSE}(\hat{f}, \mathbf{S}) = \frac{1}{|\mathbf{S}|} \sum_{i=1}^{|\mathbf{S}|} ||y_i - \hat{f}(x_i)||^2 \tag{2.3}$$

Jednym z podstawowych powodów, dla którego ta funkcja strat została użyta w niniejszej pracy jest jej duża popularność. Kolejnym powodem są jej dobre właściwości matematyczne. W szczególności: średni błąd kwadratowy jest minimalizowany przez średnią rozkładu warunkowego, która sama w sobie jest pożądaną do uzyskania statystyką. Ponadto, jego pochodna ze względu na wynik aproksymacji istnieje dla całej użytej przestrzeni oraz jest funkcją liniową, co ułatwia stosowanie metod ją wykorzystujących, np. gradientowych. Proporcjonalność tej funkcji strat do kwadratu odchylenia sprawia, że szczególnie niepożądane są duże błędy. Jest to cecha użyteczna w wielu praktycznych przypadkach predykcji własności funkcji.

Miarą, która karze błędy proporcjonalnie do ich rozmiaru, jest średni błąd na moduł, oznaczany jako E_{abs}.

$$E_{abs}(\hat{f}, \mathbf{S}) = \frac{1}{|\mathbf{S}|} \sum_{i=1}^{|\mathbf{S}|} ||y_i - \hat{f}(x_i)||_1 \tag{2.4}$$

gdzie $||\cdot||_1$ to norma l^1 ("Manhattan"). W rozdziałach 5 i 6, opisujących eksperymenty, używany jest także pierwiastek ze średniego błędu kwadratowego (*root*

mean squared error, RMSE) i jego wersje znormalizowane.

Generalizacja i uczenie

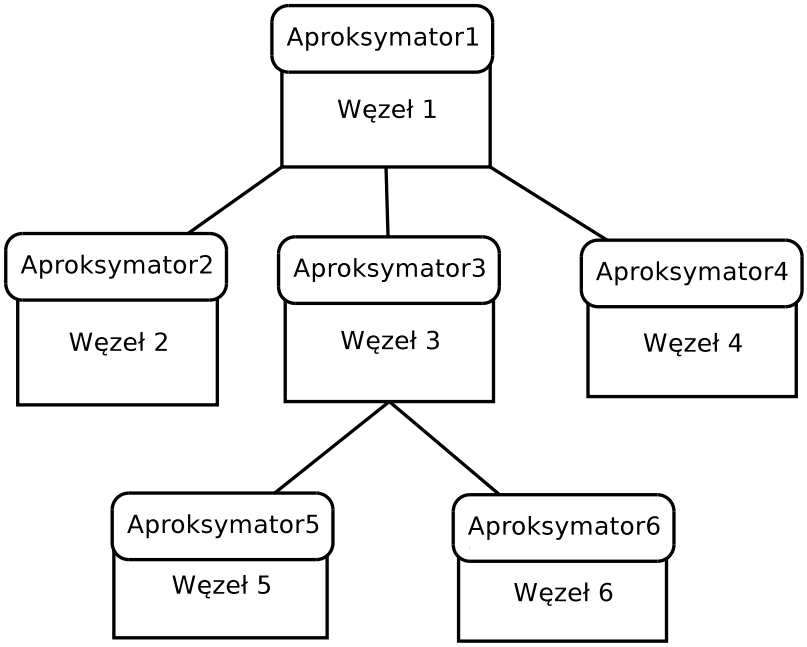
Fakt, że zmienna wyjściowa nie musi zależeć w sposób całkowicie jednoznaczny od zmiennych wejściowych sprawia, że błąd na zbiorze testowym równy 0 dla aproksymacji funkcji jest praktycznie nie do osiągnięcia. Ponadto, fakt ten również zmniejsza wiarygodność testowania. W niniejszej pracy założono, że przy odpowiednio dużych i rozsądnie dobranych próbkach, można wyniki tegoż testowania uznać za miarodajne (za np. [48]).

Z kolei błąd bliski zeru na zbiorze uczącym może nie być pożądany. Jeżeli zmienności wyjścia nie da się w pełni wyjaśnić zmiennością wejścia, a rozwiązanie – jak to proponowane w rozprawie – opiera się na dopasowywaniu funkcji na zbiorze uczącym, bardzo dobre odtworzenie par wejście-wyjście na zbiorze uczącym może skutkować dopasowaniem *nadmiernym*. Jest to sytuacja, w której pomimo zmniejszenia funkcji strat na zbiorze uczącym, funkcja ryzyka rośnie, co zwykle wywołuje wzrost błędu na zbiorze testującym. Oznacza ona niską *generalizację*. Większe ryzyko nadmiernego dopasowania pojawia się w systemach o większej mocy, tzn. będących w stanie przybliżyć szerszą klasę funkcji [94]. Zależy ono również od tego, jak bliskie przybliżenie *prawdziwej* relacji jest w stanie uzyskać dany system.

Podejście konstruktywne, prezentowane w niniejszej dysertacji, można rozpatrywać jako systematyczne rozszerzanie dopasowywanej klasy funkcji, o ile w poprzedniej nie znaleźliśmy odpowiedniego rozwiązania. Jako że w utworzonym już modelu znajduje się niebagatelna ilość informacji o problemie, model ten pozostaje częścią nowego, a jednocześnie w dużej mierze steruje rozszerzaniem – powstaje struktura hierarchiczna. Rozszerzanie to dokonywane jest stopniowo, w każdym kroku dodawana jest aproksymacja rozwiązująca pewien podproblem całości. Jako najmniejszy krok rozszerzania pozostaje ona względnie prosta, by uniknąć nadmiernego dopasowania. Aby otrzymać dokładną aproksymację, wyniki tych aproksymacji są łączone w średniej ważonej (ze zmiennymi wagami w zależności od przykładu).

Wspomniana już hierarchiczność rozwiązania odbija się też w strukturze podproblemów. Skoro najpierw tworzone są rozwiązania ogólne, a na ich podstawie tworzone są części bardziej szczegółowe - bardziej lokalne, obejmujące tylko wycinek całego problemu, to podproblemy również wykazują pewną hierarchiczną strukturę. Wynika to z założenia, że przy ograniczonej pojemności modeli składowych, pewne podproblemy mogą być łatwiejsze do nauczenia z osobna niż jako część większego problemu. Podproblemy te nie są całkowicie rozłączne, gdyż, jak zostało wykazane w twierdzeniach z rozdziału 3 oraz pracach wspomnianych w podrozdziale 1.4 (np.

[42, 46, 76]), nakładanie się podproblemów może ograniczać błąd. Dla systemu hierarchicznego, w którym części utworzone później mają mniejszy zakres działania niż utworzone wcześniej, naturalną wydawała się struktura drzewa. Co więcej, rozłączne gałęzie drzewa mogą być uczone w sposób niezależny, co ułatwia implementację równoległą algorytmu oraz upraszcza zadanie podziału problemu (dodając wymaganie, że podproblem danej części powinien zawierać się w podproblemie jego rodzica w drzewie). Przykładowa struktura pokazana jest na rysunku 2.1.



Rysunek 2.1. Przykładowa struktura drzewa rozwiązania.

2.2. Podstawowe definicje rozwiązania

Rozwiązanie opisane w niniejszej pracy – hierarchiczna aproksymacja – w sensie najbardziej abstrakcyjnym można określić jako specyficzną klasę funkcji przypisujących zestawowi parametrów kontrolnych oraz zbiorowi uczącemu inną funkcję nazywaną dalej nauczonym (lub gotowym) rozwiązaniem bądź nauczoną hierarchiczną aproksymacją. Słowo "nauczony" może czasami być opuszczane, o ile kontekst jest jednoznaczny. Jest to rozwiązanie ramowe, używające gotowych rozwiązań, np. sieci neuronowych lub algorytmów grupowania jako niektórych ze swych części, mogące rozwiązań tych używać wymiennie, choć zmiany w tym zakresie mogą mieć wpływ na błędy metody.

Własności definiujące proponowanego rozwiązania będą opisywane przede wszystkim w sposób algorytmiczny – jako meta-algorytm uczenia maszynowego (zob. 2.4.2) i jego uszczegółowienia (rozdz. 4).

Nauczona hierarchiczna aproksymacja jest funkcją z przestrzeni wejść do przestrzeni wyjść $HA : \mathcal{X} \longrightarrow \mathcal{Y}$ o opisanych niżej własnościach.

Przede wszystkim, jest ona oparta na, tworzonej w czasie uczenia, strukturze drzewa - być może różnej dla różnych zadań. W dalszej części pracy będzie używane skrótowe określenie "węzeł i" na oznaczenie węzła drzewa o indeksie i.

Niech $N(i)$ będzie oznaczeniem liczby dzieci węzła i.

Definicja 1 (Składniki węzła)**.** Każdy węzeł utworzonego (nauczonego) drzewa zawiera:

1. Prostą aproksymację szukanej funkcji, określoną na pewnym fragmencie przestrzeni wejść. Oznaczana będzie jako $g_i : \mathcal{X}_i \to \mathcal{Y}_i$, gdzie $\mathcal{X}_i$ to fragment przestrzeni wejść, na którym g_i jest określona, a $\mathcal{Y}_i$ jej obraz. Aproksymacja ta będzie rozwiązaniem pewnego podproblemu problemu głównego – w korzeniu całego problemu. Ma ona być prosta w sensie stosunkowo małej mocy, w celu uniknięcia nadmiernego dopasowania i zmniejszenia czasu nauki. Podstawowymi rodzajami użytych aproksymatorów są proste jednokierunkowe sieci neuronowe (typu *feed-forward*) nauczane odmianami algorytmu wstecznej propagacji błędu – por. podrozdział 4.3.
2. Funkcję *kompetencji* $C_i : \{0, \ldots, N(i)\} \times \mathcal{X} \longrightarrow [0, 1]$, gdzie $C_i(0)$ oznacza funkcję kompetencji rozwiązania składowego umieszczonego w i-tym węźle. Służy ona jako funkcja wag przy uzyskiwaniu odpowiedzi rozwiązania według równania (2.6) przedstawionego poniżej. Jej podstawą jest oszacowanie przydatności danego węzła do oceny danego przykładu. Może występować w kilku wersjach, które są opisane w pracach autora rozprawy [23, 24] oraz rozdziale 4 rozprawy. W zależności od wersji, przy obliczaniu tej funkcji wektor cech (x) może być użyty w sposób bezpośredni lub pośredni – np. poprzez wartość wyjścia przypisaną mu przez aproksymację w i-tym węźle.
   Dla każdego przykładu z $\mathcal{X}_i$ suma funkcji kompetencji dla wszystkich bezpośrednich potomków i rozwiązania w danym węźle wynosi 1:

$$\forall x_k \sum_{j=0}^{N(i)} C_i(j, x_k) = 1 \tag{2.5}$$

W przypadkach, gdy mowa jest o wartości danej aproksymacji lub całego rozwiązania na jakimś zbiorze przykładów, oczywiście oznacza to wartości dla wektorów

cech tych przykładów.

Do opisu drzewa przydatna będzie następująca notacja: węzły będące bezpośrednimi potomkami danego węzła są ponumerowane kolejnymi liczbami naturalnymi od 1 do $N(i)$, a funkcja $I : \mathbf{I} \times \mathbb{N} \to \mathbf{I}$ jest zdefiniowana w ten sposób, że $I(i, j)$ oddaje unikalny indeks j-tego węzła potomnego węzła i (zbiór indeksów oznaczany jako $\mathbf{I}$). Dla wygody zapisu, za węzeł potomny 0 uważa się aproksymację w i-tym węźle (ściślej: "wirtualny" liść zawierający tą aproksymację). Przypisanie rodzica do węzła potomnego oznaczane jest przez R ($R : \mathbf{I} \to \mathbf{I}$).

Odpowiedź nauczonego rozwiązania dla *danego węzła* i k-tego przykładu jest oznaczana za pomocą $\widetilde{g}_i(x_k)$ a otrzymywana jest w następujący sposób:

Definicja 2 (Wynik węzła)**.**

$$\widetilde{g}_i(x_k) = \sum_{j=1}^{N(i)} \widetilde{g}_{I(i,j)}(x_k) \cdot C_i(j, x_k) + C_i(0, x_k) \cdot g_i(x_k) \tag{2.6}$$

Dla uproszczenia zapisu oznaczamy g_i jako $\widetilde{g}_{I(i,0)}$, wtedy powyższy wzór przyjmie postać:

$$\widetilde{g}_i(x_k) = \sum_{j=0}^{N(i)} \widetilde{g}_{I(i,j)}(x_k) \cdot C_i(j, x_k)$$

Odpowiedzią rozwiązania jest odpowiedź korzenia.

Funkcja kompetencji jest podstawowym narzędziem realizacji koncepcji łączenia wyników wielu predykcji w jedną, potencjalnie o mniejszym błędzie.

Istnieje jednak wiele sposobów łączenia wyników w węźle. Arytmetyczna średnia ważona została wybrana przede wszystkim ze względu na jej prostotę oraz pewien kompromis pomiędzy elastycznością, a odpornością na nietypowe odpowiedzi węzłów. Co więcej - pewne wyniki teoretyczne, pokazane poniżej (rozdz. 3), wymagają takiego sposobu łączenia, a użycie ich jest wynikiem przyjęcia błędu średniokwadratowego jako podstawowej funkcji kosztu. Kolejnym powodem wyboru takiego sposobu łączenia wyników jest relatywna stabilność numeryczna i efektywność obliczania (np. w porównaniu do ważonej sumy geometrycznej). Powody, dla których średnia (czyli nieujemne współczynniki sumujące się do 1) nie została zastąpiona sumą ważoną, są dwa. Po pierwsze, średnia ważona – w odróżnieniu od sumy – gwarantuje, że błąd całości nie będzie większy niż największy z błędów tych aproksymacji, które otrzymały niezerową funkcję kompetencji. Drugim powodem jest to, że skoro wszystkie składowe mają być przybliżeniami tej samej szukanej wartości, zmiana skali za pomocą funkcji wag lub przypisywanie wag ujemnych nie posiada

dobrej interpretacji i wprowadza dodatkowe skomplikowanie do systemu. Z kolei, w stosunku do zwykłej średniej lub np. mediany, średnia ważona ma większą elastyczność i zdolność do wykorzystania różnic nawet przy bardzo niedokładnych przewidywaniach błędów poszczególnych aproksymacji.

Istotnymi pojęciami używanymi w dalszym ciągu pracy są *zbiór kompetencji* i *przestrzeń kompetencji*.

Definicja 3 (Przestrzeń kompetencji)**.** Przestrzenią kompetencji nazywamy zbiór wszystkich przykładów, które mogą kiedykolwiek być poddane ocenie danego węzła.

Dla korzenia są to wszystkie dozwolone przykłady: $(x, y) : x \in \mathcal{X}, y \in \mathcal{Y}$.

Dla węzłów potomnych są to takie przykłady (wejścia), dla których funkcja kompetencji (wag) C rodzica przyjmuje nieujemne wartości – dla węzła rodzica i oraz jego j-tego węzła potomnego: $(x, y) : C_i(j, x) > 0$.

Definicja 4 (Zbiór kompetencji)**.** Zbiór kompetencji danego węzła w danym zbiorze przykładów to przecięcie danego zbioru przykładów oraz przestrzeni kompetencji danego węzła. Wspomniany *dany zbiór* nie jest, niekiedy, wyszczególniony explicite, ale znany z kontekstu.

Istotną cechą odróżniającą dane rozwiązanie np. od *Hierarchical Mixture of Experts* (rozdz. 1.4, [53]) jest fakt, że w każdym węźle, nie tylko w liściach, znajduje się rozwiązanie pewnego podproblemu i jego odpowiedzi jak najbardziej mogą wejść w skład odpowiedzi całości rozwiązania. Głównym celem takiego podejścia jest zwiększenie stabilności rozwiązania. Aproksymacje w węzłach wewnętrznych mają do rozwiązania większe części problemu i otrzymują większe liczby przykładów uczących niż niż aproksymacje w liściach. Zazwyczaj są przez to mniej dokładne od aproksymacji w węzłach potomnych (liściach), ale też nie wykazują nadmiernego dopasowania. Fakt, że wewnętrzne węzły drzewa również zwracają przewidywaną wartość wyjścia dla danego wejścia, zwiększa swobodę doboru zbiorów kompetencji dzieci, które dzięki temu nie muszą bezwzględnie sumować się dokładnie do przestrzeni kompetencji rodzica. Jest on również przydatny, gdy dany przykład ocenia jedynie jeden węzeł potomny, gdyż, zgodnie z twierdzeniami z rozdziału 3, możliwość uwzględnienia wyniku więcej niż jednego rozwiązania składowego dla danego przykładu jest zasadniczo korzystna, szczególnie, kiedy aproksymacje w poszczególnych węzłach wykazują cechy losowości przy tworzeniu, jak ma to miejsce w losowo inicjalizowanych sieciach neuronowych.

Korzystając ze schematu tworzenia drzewa, opisanego w podrozdziale 2.4.2, w chwili tworzenia danego węzła nie można być pewnym, czy będzie on liściem, czy węzłem wewnętrznym, ponieważ struktura drzewa dostosowuje się automatycznie, w sposób

przede wszystkim przyrostowy, do danego zadania – co jest jedną z głównych zalet rozwiązania. Aproksymacje w węzłach wewnętrznych muszą być więc na pewnym etapie i tak utworzone, więc ich zachowanie nie powoduje zwiększenia czasu uczenia.

2.3. Algorytm uzyskiwania wyniku

Algorytm 1 (Uzyskiwania wyniku)**.** Ze względu na strukturę proponowanego rozwiązania, algorytm uzyskiwania wyniku opisany zostanie za pomocą rekurencji. Węzeł w którym wykonywany jest algorytm, jest nazywany *węzłem aktywnym*, jego indeks oznaczamy przez i.

1. Ustaw korzeń jako *węzeł aktywny*.
2. Uzyskaj predykcję wartości funkcji z użyciem prostej aproksymacji funkcji w węźle aktywnym (g_i).
3. Oblicz wartości funkcji kompetencji dla danego wejścia x i wszystkich dzieci węzła aktywnego używając odpowiedzi tego węzła, w tym jego błędów i ewentualnie wektora cech.
4. Dla każdego węzła potomnego, dla którego wartość funkcji kompetencji ($C_i(j, x)$) jest większa niż 0, ustaw go jako *węzeł aktywny* i uruchom dla niego rekursywnie algorytm od punktu 2.
5. Oblicz odpowiedź *węzła aktywnego* używając równania (2.6).

Odpowiedzią drzewa jest odpowiedź korzenia, zgodnie z definicją 2, s. 18. Jak widać, jest to ogólny schemat, w którym należy przede wszystkim doprecyzować, jakie algorytmy zostaną wywołane w punktach 2 i 3. Sposoby otrzymywania funkcji kompetencji dla punktu 3, opisane w rozdziale 4, są oryginalne, natomiast zasadniczo dla składowej aproksymacji stosowane są istniejące rozwiązania - proste sieci neuronowe, jednak z automatycznym (heurystycznym i dość prostym) doborem architektury sieci.

Algorytm dla danego wejścia x i danej odnogi drzewa kończy się, jeżeli nie zostanie znaleziony żaden j-ty węzeł potomny o $C_i(j, x) > 0$, a więc, zgodnie z warunkiem (2.5), jeżeli $C_i(0, x) = 1$. Jest to zawsze spełnione w liściu, ale dla poszczególnych przykładów może być też spełnione w węźle wewnętrznym.

Wektor wejściowy jest najpierw rozpowszechniany w dół drzewa, ale tylko wzdłuż niektórych ścieżek – tych, gdzie odpowiednie funkcje kompetencji są większe od 0. Następnie wyniki z poszczególnych węzłów są zbierane w górę drzewa i uśredniane z wartościami funkcji kompetencji jako wagami. W ogólnym schemacie działania nie ma przeciwwskazań do użycia całego drzewa dla konkretnego przykładu, jednak

w praktyce unika się takiej sytuacji ze względu na konstrukcję funkcji kompetencji (zob. podrozdział 4.1).

Należy też zauważyć, że dla danego przykładu odpowiedź rozwiązania jest średnią ważoną odpowiedzi poszczególnych aproksymatorów w drzewie. Waga odpowiedzi danego węzła dla danego przykładu wynosi $\prod_{k=0}^{gl.-1} C_{i_k}(j_k, x)$, gdzie i_k to indeksy węzłów na ścieżce od korzenia do danego węzła: $i_k = R(i_{k+1})$, $i_{k+1} = I(i_k, j_k)$, i_0 to korzeń, $gl.$ oznacza głębokość, $I(i_{gl.-1}, j_{gl.-1})$ to indeks danego węzła. Tak prosta zależność zachodzi tylko dla pojedynczego przykładu, natomiast aproksymacja wynikowa nie jest kombinacją liniową aproksymatorów w drzewie dzięki zmienności wag C_i względem przykładu.

2.4. Algorytm uczenia

Powyższy schemat uzyskiwania wyniku wymaga już nauczonej hierarchicznej aproksymacji, o której poniżej.

2.4.1. Podstawowe cele uczenia

Nauczona hierarchiczna aproksymacja jest drzewem o specyficznych elementach w węźle (def. 1). Algorytm uczenia powinien zatem znaleźć:

1. strukturę drzewa i podział na podproblemy, który według pewnych kryteriów jest odpowiedni dla danego problemu,
2. dokładne funkcje kompetencji dla każdego węzła,
3. aproksymacje funkcji w każdym węźle.

Cel 1 w szczególności oznacza znalezienie liczby dzieci każdego węzła. W istniejących wersjach metody cele 1 i 2 wykonywane są łącznie. Należy zauważyć, że samo utworzenie zbiorów uczących można traktować zarówno w kategoriach podziału na podproblemy jak i podstawowej części uzyskiwania aproksymacji w danym węźle.

Dla krótszego opisu takiego podziału wprowadzimy dodatkowe oznaczenie - funkcję podziału danych uczących U.

Definicja 5. W czasie uczenia każdemu węzłowi i przypisywana jest funkcja podziału danych uczących: $U_i : \mathbb{N} \times \mathcal{X} \times \mathcal{Y} \longrightarrow [0, 1]$ taka, że

$$U_i(j, x, y) > 0 \Longleftrightarrow (x, y) \in \mathbf{U}_{I(i,j)}$$

gdzie $\mathbf{U}_{I(i,j)}$ jest zbiorem uczącym j-tego dziecka węzła i.

Jeżeli dana metoda aproksymacji funkcji wspiera ważenie przykładów wejściowych, wartości tej funkcji powinny dać się użyć jako wagi.

Progowanie funkcji U daje funkcję charakterystyczną zbioru uczącego. Jej wartości istotne są tylko dla przykładów ze zbioru uczącego rodzica, ponieważ tylko z nich mogą pochodzić przykłady w zbiorach uczących dzieci.

Podobnie jak działanie, uczenie w tej części jest opisane w formie schematu algorytmu. Może on być konkretyzowany na wiele sposobów, niektóre z nich są opisane w dalszej części dysertacji (por. rozdz. 3).

2.4.2. Schemat algorytmu uczenia

Wejściem systemu jest zbiór danych uczących $\mathbf{U}$ oraz parametry kontrolne. Podstawowymi parametrami kontrolnymi są E_{min} - określający "docelowy" błąd w jednym węźle oraz ograniczenie na wysokość drzewa. Alternatywnie, zamiast lub obok drugiego parametru kontrolnego można podać inne ograniczenia, np. na czas uczenia lub na sumaryczną liczność zbiorów uczących wszystkich aproksymacji.

Węzeł, którego dotyczą punkty algorytmu, będzie nazywany *węzłem aktywnym.*

Algorytm 2 (Schemat uczenia)**.**

1. Utwórz korzeń i przypisz mu cały zbiór $\mathbf{U}$ jako zbiór uczący oraz całą przestrzeń $\mathcal{X}$ jako przestrzeń kompetencji. Wstaw go do kolejki (struktury) $\mathbf{Q}$.
2. Pobierz kolejny węzeł z $\mathbf{Q}$, będzie on teraz *węzłem aktywnym* (oznaczmy jego indeks jako i). Utwórz w tym węźle, za pomocą metod uczenia maszynowego, prostą aproksymację poszukiwanej funkcji na jego zbiorze uczącym (g_i). Rozwiązanie zastosowane tutaj powinno być dość proste, np. sieć neuronowa z kilkoma neuronami ukrytymi. Punkt ten dokładniej zostanie opisany w podrozdziale 4.2.
3. Oblicz miarę błędu tej aproksymacji na jej zbiorze kompetencji: $E_{abs}(g_i, \mathbf{S}_i)$. W niniejszej pracy jest to błąd bezwzględny, ale nie jest to jedyna możliwość. Jeżeli $E_{abs}(g_i, \mathbf{S}_i) < E_{min}$ należy zakończyć ten algorytm dla tego węzła bez tworzenia węzłów potomnych. Należy zaznaczyć, że $\mathbf{S}_i$ nie musi być identyczny ze zbiorem uczącym – może być jego podzbiorem.
4. Dla każdego węzła niebędącego korzeniem, porównaj jego błąd średniokwadratowy na zbiorze kompetencji z błędem jego rodzica na *tym samym zbiorze.* Jeżeli jest większy niż analogiczny błąd rodzica, a jest używana aproksymacja o charakterze stochastycznym (np. losowa inicjalizacja wag sieci neuronowej), utwórz aproksymację ponownie, o ile nie powoduje to złamania warunku z punktu 5. Jeżeli druga aproksymacja wciąż cechuje się większym błędem, zakończ algorytm

dla tej gałęzi (tego węzła), ale ponadto usuń aktywny węzeł z drzewa.

5. Jeżeli osiągnięty został maksymalny poziom złożoności drzewa podany jako parametr algorytmu, zakończ algorytm dla tej odnogi. Maksymalny stopień złożoności może być wyrażany w różny sposób, w eksperymentach jest to głębokość drzewa.
6. Utwórz
   a) Zbiory uczące dla węzłów potomnych $\{\mathbf{U}_{I(i,1)} \dots \mathbf{U}_{I(i,N(i))}\}$, konstruując funkcję podziału danych U_i, taką że $(x_k, y^{(k)}) \in \mathbf{U}_{I(i,j)} \Leftrightarrow U_i(j, x_k, y^{(k)}) > 0$; zbiory uczące mają części wspólne, nie muszą też sumować się do całości zbioru uczącego rodzica – choć w praktyce taka sytuacja jest bardzo rzadka; liczba węzłów potomnych ($N(i)$) może zostać ustalona lub znaleziona.
   b) Funkcję kompetencji C_i.
7. Wstaw węzły potomne do struktury $\mathbf{Q}$ i wywołaj algorytm od punktu 2.

Zadania 6a i 6b są rozwiązywane łącznie. Są one na tyle istotne, że warianty metod tworzenia tych funkcji zostały wyodrębnione w rozdziale 4. Ich tworzenie opiera się przede wszystkim na analizie odpowiedzi rozwiązania składowego węzła rodzica, prawdziwych wartościach ze zbioru uczącego oraz na ich połączeniu, czyli błędach aproksymacji. Bardzo często używane są tu formy grupowania. Podstawy teoretyczne opisane w rozdziale 3 również w dużym stopniu służą jako wskazówki do tworzenia konkretnych realizacji tego kroku.

Najistotniejszym z punktu widzenia niniejszej pracy krokiem tego meta-algorytmu jest punkt 6a (opisany szczegółowo w podrozdziale 4.1). Jego dokładna realizacja ma decydujący wpływ na strukturę drzewa i duży wpływ na błąd (rozdz. 5, porównanie z wersjami proponowanego rozwiązania opisanymi przez autora niniejszej pracy w artykułach [23, 24]). Kolejny punkt wymagający sprecyzowania to punkt 2. Oczywiście, jego wpływ na wielkość błędu jest również duży, jednak w zasadzie jest on odpowiednim połączeniem istniejących metod.

W punkcie 2 może co do zasady zostać użyty dowolny aproksymator funkcji, choć zapewne nie wszystkie dadzą takie same rezultaty. W niniejszej pracy założono, że powinien on być stosunkowo prosty, by ograniczyć ryzyko nadmiernego dopasowania. Szczegóły aktualnie używanego wariantu znajdują się w podrozdziale 4.2.

Dwa podstawowe warunki zatrzymania algorytmu uczącego (punkty 3 i 4) mogą powodować, że liście znajdują się na różnej głębokości w drzewie. Sposoby znajdowania funkcji kompetencji i zbiorów uczących węzłów potomnych mogą powodować zmienność liczby węzłów potomnych, pewien wpływ na to mają także metody tworzenia rozwiązań składowych w węźle (ponieważ używane są jego predykcje i błędy).

Połączenie tych dwóch okoliczności skutkować może stosunkowo dużą zmiennością struktury drzewa w zależności od zaprezentowanego zbioru uczącego. W tym sensie można stwierdzić, że rozwiązanie dopasowuje strukturę drzewa do rozwiązywanego zadania.

Warunki zatrzymania wymagają pewnej znajomości problemu – przynajmniej jakiego rzędu wielkości błąd jest oczekiwany, choć eksperymenty nie wykazały, aby było to absolutnie konieczne (rozdz. 5 i 6). Drugi parametr dotyczy ograniczenia na koszt obliczeniowy utworzenia drzewa i może być, w sposób szacunkowy, przeliczony na czas uczenia (zwłaszcza, jeżeli użyte są rozwiązania składowe o liniowym czasie nauki względem rozmiarów zbiorów uczących). W istocie obydwa parametry dotyczą częściowo dylematu czas uczenia – dokładność: określenia przez użytkownika algorytmu tego, co chce osiągnąć (błąd) i jakim maksymalnym kosztem. W innych aspektach rozwiązanie to stara się być w pełni automatyczne, ponieważ jego zastosowaniem mają być właśnie te części systemów sztucznej inteligencji, do których nie jesteśmy w stanie dostarczyć wiele więcej niż przykłady uczące. Wspomnianego kompromisu w zasadzie nie sposób jednak uniknąć. Zwykle, mając do dyspozycji większą ilość czasu, system może się nauczyć lepiej (do pewnych granic), ale często lepiej mieć trochę gorszy wynik na czas niż lepszy po czasie. Samo rozwiązanie, ze względu na prostotę rozwiązań składowych w węzłach, wykazuje przynajmniej częściową odporność na przeuczenie (por. rozdz. 5 i 6). Odporność ta ma jednak swoje granice: ustawienia nierealistycznie małych błędów, połączone z możliwością osiągnięcia dużych wysokości drzewa, mogą spowodować przeuczenie i bardzo długą naukę.

Struktura **Q** w obecnym rozwiązaniu również może mieć różne warianty. Podstawowym jest kolejka priorytetowa ze względu na głębokość węzła – węzeł o niższej głębokości ma wyższy priorytet. W opisanym przypadku klasyfikatory są tworzone w porządku *pre-order*. Ponieważ zasadniczo dwie gałęzie są niezależne od siebie, zastosowano implementację równoległą (zob. podrozdział 5.3). W przypadku używania innych ograniczeń na złożoność drzewa niż jego głębokość, dokładna kolejność tworzenia ma duże znaczenie, potencjalnie najlepszym jest porządek BFS, czyli "najpierw wszerz".

Podstawową cechą tego rozwiązania jest podział problemu na podproblemy. Po uwzględnieniu algorytmu uczącego można dostrzec ogólny zarys sposobu w jaki jest to osiągane. Dzielone są zarówno obszary kompetencji – węzeł może dostać do oceny tylko przykłady z pewnego zakresu, jak i zbiory uczące dla węzłów potomnych, będące z zasady (a w niektórych rozwiązaniach z definicji – podrozdział 4.1) mniejsze niż zbiór rodzica. Powiązane jest to z nadzieją, że problem, który jest w jakiś istotny

sposób ograniczony, będzie łatwiejszy (w sensie miary błędu) do nauczenia. Problem nie może być jednak ograniczony za bardzo, bo wtedy węzeł może nie zostać nigdy lub prawie nigdy użyty i uczenie będzie nieefektywne. Ograniczenie to nie powinno polegać na wyrzuceniu zupełnie losowych przykładów uczących, gdyż może to spowodować brak istotnych informacji. W niniejszej pracy ta koncepcja jest domeną heurezy pojawiającej się w różnych wariantach w zależności od wersji rozwiązań szczegółowych (rozdz. 4).

3. Teoretyczne oszacowania błędu proponowanej metody

W tym rozdziale przedstawione zostaną wyniki teoretyczne dotyczące schematu (meta-algorytmu) hierarchicznego aproksymatora. Zostaną one przedstawione przed opisem szczegółowych wariantów algorytmu (podrozdział 4.1) ponieważ od tych wariantów nie zależą, wręcz przeciwnie: opisując składowe błędu dla danego węzła i całego rozwiązania są jedną z podstaw tworzenia owych wariantów.

Oszacowania teoretyczne a testowanie

Oszacowania teoretyczne, na podstawie matematycznych własności rozwiązań, dają wyniki pewne (choć ta pewność może np. dotyczyć wartości pewnego prawdopodobieństwa), ale kosztem często bardzo trudnej adaptacji do konkretnych rozwiązań (np. oszacowania używające wymiaru Vapnika-Chervonenkisa [94]). Oszacowania górne powstałe w ten sposób, użyte do określenia generalizacji mogą być nadmiernie pesymistyczne. Wydaje się to być konsekwencją połączenia dokładnego oszacowania z niemożnością uwzględnienia wszystkich cech specyficznych danego problemu (których poznanie zresztą może nie być wiele łatwiejsze od rozwiązania zadania), skutkującego pewnego rodzaju maksimum błędu po wszystkich możliwych problemach w danej klasie.

Dlatego podstawową "praktyczną" metodą oszacowywania funkcji ryzyka oraz jakości rozwiązania, używaną również w niniejszej pracy, jest testowanie – obliczenie funkcji strat na pewnej próbce rzeczywistych danych.

Podstawową zaletą testowania jest uwzględnienie właściwości problemu. Jednak ma też ona pewne wady. Ciąg testowy powinien być jak najbardziej reprezentatywny dla problemu, który ma być rozwiązany. Skoro zaś dany problem nie jest jeszcze rozwiązany (a jeśli zaszła konieczność uczenia maszynowego, to być może nawet nie jest dogłębnie poznany), to stwierdzenie z całkowitą pewnością reprezentatywności próbki w zasadzie nie jest możliwe. Liczba przykładów testowych jest ograniczona, więc zwykle będzie ona reprezentatywna dla pewnej części problemu. Żeby zatem funkcja strat obliczona na zbiorze testowym była użyteczną miarą ryzyka, wymagane jest, aby między zbiorem testującym, a testowanym systemem było jak najmniej

zależności poza tą, że oba dotyczą całości problemu.

W przypadku systemów używających danych w więcej niż jeden sposób (np. również do tworzenia modelu jak systemy tworzenia sieci neuronowych, czy też zwykła selekcja sieci neuronowych) dla zachowania wiarygodności testowania trzeba pamiętać o wykluczeniu danych testowych z każdego etapu powstawania rozwiązania.

W tym rozdziale dokonano pewnych oszacowań teoretycznych, przede wszystkim opisujących strukturę błędu, a w rozdziałach 5 i 6 wykonano testy na rzeczywistych zadaniach uczących.

3.1. Oznaczenia dotyczące twierdzeń

Na początku zostaną przedstawione dodatkowe oznaczenia, które będą przydatne przy formułowaniu twierdzeń. W podrozdziale 2.1 został wspomniany błąd średniokwadratowy i błąd bezwzględny. Dla wygody błąd kwadratowy aproksymacji $\hat{f}$ dla jednego przykładu (x_k, y_k) będzie oznaczany jako:

$$e(\hat{f}, x_k) = \sum_{l=1}^{r} (\hat{f}(x_k)_l - y_{kl})^2 \tag{3.1}$$

W wielu przypadkach przydatna będzie też uproszczona notacja dla wektora odchyleń w danym węźle i:

$$\eta_i(x_k) = g_i(x_k) - y_k \tag{3.2}$$

$$\tilde{\eta}_i(x_k) = \widetilde{g}_i(x_k) - y_k \tag{3.3}$$

Ze względu na to, że często operuje się na odchyleniach węzłów potomnych oraz brane są poszczególne współrzędne odchyleń, oznaczenie to występuje często w postaci: $\tilde{\eta}_{I(i,j)}(x_k)_l$.

Błąd średniokwadratowy można zapisać jako:

$$\mathrm{MSE}(\widetilde{g}_i, \mathbf{S}) = \frac{1}{|S|} \sum_{x_k, y_k \in S} ||\tilde{\eta}_i(x_k)||^2 \tag{3.4}$$

także

$$e(\widetilde{g}_i, x_k) = ||\tilde{\eta}_i(x_k)||^2 \tag{3.5}$$

Cosinus kąta między dwoma wektorami oznaczony jest przez $\cos(\measuredangle \overrightarrow{X}, \overrightarrow{Y})$. Natomiast $[x_i]_{i=0}^{n}$ oznacza wektor o długości n i współrzędnych $x_1, \ldots, x_n$.

3.2. Błędy na pojedynczym przykładzie

Głównym celem zamieszczonych tu lematów jest rozłożenie błędów dla każdego przykładu na czynniki poddające się interpretacji i dające podstawę do dalszych rozważań.

Lemat 1 (Podstawowy o błędzie dla przykładu)**.** *Dla dowolnego przykładu x_k i dowolnego węzła i, błąd kwadratowy węzła: $e(\widetilde{g}_i, x_k)$ jest postaci:*

$$e(\widetilde{g}_i, x_k) = \sum_{l=1}^{r} \sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot C_i(j, x_k) \cdot \cdot \cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{C_i(j, x_k)} \right]_{j=0}^{N(i)}, \left[\sqrt{C_i(j, x_k)} \right]_{j=0}^{N(i)} \right) \tag{3.6}$$

$$= \sum_{l=1}^{r} \sum_{j: C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot C_i(j, x_k) \cdot \cdot \cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{C_i(j, x_k)} \right]_{j: C_i(j,x_k)>0}, \left[\sqrt{C_i(j, x_k)} \right]_{j: C_i(j,x_k)>0} \right) \tag{3.7}$$

Dowód znajduje się w dodatku A.

Jak widać błąd kwadratowy sumowany po współrzędnych został rozłożony na sumę średnich ważonych błędów kwadratowych węzłów potomnych i aproksymacji w węźle na poszczególnych współrzędnych z funkcją kompetencji jako wagą, pomnożonych przez kwadrat cosinusa kąta pomiędzy dwoma ustalonymi wektorami. Jest to o tyle pożądane, że drugi czynnik jest zawsze mniejszy lub równy 1, a co za tym idzie, błąd na danym przykładzie jest zawsze mniejszy lub równy średniemu błędowi kwadratowemu wybranych węzłów, ważonemu funkcją kompetencji:

Wniosek 1 (O nierówności dla przykładu)**.**

$$e(\widetilde{g}_i, x_k) \leqslant \sum_{l=1}^{r} \sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot C_i(j, x_k) \tag{3.8}$$

Co więcej, równość zachodzi w dość specyficznym przypadku, stąd błąd ten jest zwykle mniejszy, o czym mówi następny wniosek.

Wniosek 2 (O ścisłej nierówności dla przykładu)**.**

$$e(\tilde{g}_i, x_k) \lneq \sum_{l=1}^{r} \sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot C_i(j, x_k) \iff \exists_{j,m:C_i(j,x_k)>0, C_i(m,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k) \neq \tilde{\eta}_{I(i,m)}(x_k) \tag{3.9}$$

Błąd kwadratowy jest zatem ściśle mniejszy od średniej błędów kwadratowych użytych estymacji, jeśli tylko błędy tych estymacji się różnią. Należy zauważyć, że dotyczy to równości wektorów $\tilde{\eta}$ po współrzędnych. Wymagane jest także, aby funkcja kompetencji była niezerowa przynajmniej dla dwóch aproksymacji.

Dowód. Wynika to z faktu, że aby kąt między dwoma wektorami, z których pierwszy (w tym przypadku $\left[\sqrt{C_i(j, x_k)}\right]_{j=0}^{N(i)}$) jest zawsze niezerowy, wynosił 0, drugi z nich ($\left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{C_i(j, x_k)}\right]_{j=0}^{N(i)}$) musi być równy pierwszemu pomnożonemu przez pewną liczbę. Aby to zaszło, wszystkie współczynniki $\tilde{\eta}_{I(i,j)}(x_k)_l$ muszą być równe sobie wzajemnie tam, gdzie $\sqrt{C_i(j, x_k)} > 0$, a więc i $C_i(j, x_k) > 0$. Żeby zachować równość w sumie, musi ona zachodzić dla wszystkich współrzędnych. □

Dzięki specyficznej postaci wektorów nie ma znaczenia, czy rozpatrujemy odpowiedzi od wszystkich dzieci, czy tylko od tych, których kompetencja jest niezerowa.

Powyższe twierdzenie wskazuje, że możemy osiągnąć błąd mniejszy niż zwykła średnia ważona błędów kwadratowych. Za pewną wadę można uznać fakt, że funkcja kompetencji jest niejako wielokrotnie uwikłana z funkcją błędów, stąd też pomocny może być następujący lemat.

Lemat 2 (Alternatywny o błędzie dla przykładu)**.** *Dla dowolnego przykładu x_k i dowolnego węzła i, błąd kwadratowy węzła: $e(\tilde{g}_i, x_k)$ jest postaci:*

$$\begin{aligned} e(\tilde{g}_i, x_k) = &\sum_{l=1}^{r} \sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot \sum_{j=0}^{N(i)} C_i(j, x_k)^2 \cdot \\ &\cdot \cos^2\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l\right]_{j=0}^{N(i)}, [C_i(j, x_k)]_{j=0}^{N(i)}\right) \qquad (3.10) \\ = &\sum_{l=1}^{r} \sum_{j:C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot \sum_{j:C_i(j,x_k)>0} C_i(j, x_k)^2 \cdot \\ &\cdot \cos^2\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l\right]_{j:C_i(j,x_k)>0}, [C_i(j, x_k)]_{j:C_i(j,x_k)>0}\right) \qquad (3.11) \end{aligned}$$

Dowód znajduje się w dodatku A.

Należy tu zauważyć, że $\sum_{j:C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \leqslant \sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2$, ale zwykle nie są one równe, o ile nie zostały wybrane wszystkie dzieci węzła i: $\forall_j C_i(j,x_k) > 0$, co pociąga za sobą to, że odpowiednie kąty także zwykle nie są równe:

$$\cos^2\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l)\right]_{j:C_i(j,x_k)>0}, [C_i(j,x_k)]_{j:C_i(j,x_k)>0}\right) \geqslant$$
$$\cos^2\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l)\right]_{j=0}^{N(i)}, [C_i(j,x_k)]_{j=0}^{N(i)}\right)$$

3.3. Przykłady dla specyficznej wiedzy o błędach

Jak można wnosić z obserwacji dokonanych w podrozdziale 3.2, błąd kwadratowy węzła na danym przykładzie, oprócz liniowej zależności od błędów kwadratowych węzłów potomnych i aproksymacji w danym węźle, zależy w dużej mierze od odpowiedniego dobrania wartości funkcji kompetencji do rozkładu błędów wśród użytych aproksymacji.

Bezpośrednie dobranie funkcji kompetencji wymagałoby dokładnej wiedzy o błędach, która jednak w przypadkach najbardziej interesujących, czyli nie znajdujących się w zbiorze uczącym, jest niedostępna. O ile dla pewnych dużych próbek danych takie oszacowania mogą być dokładniejsze, o tyle oszacowania dla konkretnych przykładów musiałyby w istocie uciekać się do kolejnej regresji, co oznaczałoby przesunięcie problemu na niższy poziom zamiast jego rozwiązania.

W praktyce okazuje się, że nawet używanie bardzo zgrubnego i nieścisłego "zgadywania" daje dość dobre całościowe rezultaty (rozdz. 4, 5, 6). Metody te jednak czerpią z rozważań, które zostaną przedstawione poniżej. Są one kombinacją rozważań teoretycznych i przykładowych oszacowań numerycznych dla różnych poziomów wiedzy o błędzie. W przypadku, w którym nie mamy żadnych oszacowań błędów węzłów, z pomocą przychodzi nam najbardziej równanie (3.11). Wyrażenie, którym jesteśmy zainteresowani, ma trzy czynniki: jeden niezależny od funkcji kompetencji (suma błędów kwadratowych), jeden zależny tylko od niej (suma kwadratów funkcji kompetencji) i jeden zależny od obydwu (cosinus kąta pomiędzy odchyleniami i funkcją kompetencji podniesiony do kwadratu).

Jeśli założymy, że nie mamy wiedzy o odchyleniach i nie optymalizujemy kąta z ich udziałem, to zostaje tylko czynnik środkowy. Jest to kwadrat euklidesowej długości wektora funkcji kompetencji. Wektory, których moduły współrzędnych sumują się do 1, tworzą kulę w metryce "Manhattan". Wśród nich minimalną długość w metryce euklidesowej mają wektory postaci $\left[\pm\frac{1}{N(i)+1}\right]_{j=0}^{N(i)}$. Ponieważ funkcja kom-

petencji jest nieujemna, jedynie wektor postaci $\left[\frac{1}{N(i)+1}\right]_{j=0}^{N(i)}$ jest dopuszczalny. Jego suma kwadratów wynosi $\frac{1}{N(i)+1}$, a długość euklidesowa $\frac{1}{\sqrt{N(i)+1}}$.

Iloczyn pierwszego i drugiego składnika jest wtedy średnią arytmetyczną błędów średniokwadratowych, ale kwadrat cosinusa kąta wciąż nie musi się równać 1 – będzie tak tylko jeśli wektory odchyleń po wszystkich współrzędnych i wektor kompetencji są równoległe, a więc tylko jeśli wszystkie odchylenia mają taką samą wartość. Całkowity wynik zostanie zatem prawie zawsze polepszony. Oczekiwana wartość tej poprawy zależy od dystrybucji odchyleń.

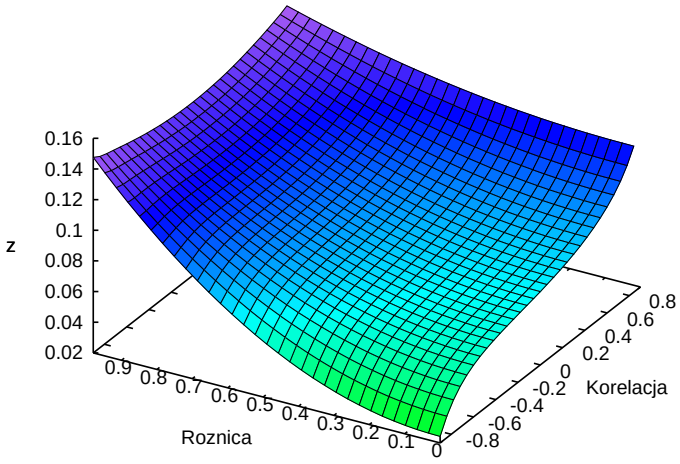
Przykład 1. Jeśli dany jest wektor na jednostkowej sferze w $\mathbb{R}^n$ (w tym przypadku wektor funkcji kompetencji), to przy jednostajnym rozkładzie kierunków drugiego wektora na sferze wartością oczekiwaną kwadratu cosinusa kąta pomiędzy tymi wektorami jest $\frac{1}{n}$.

Szczegóły uzasadnienia dla $n \geqslant 2$ znajdują się w dodatku A. W kontekście obserwacji (3.11) oznaczałoby to, że w sensie oczekiwanym błąd jest równy mniej więcej średniej błędów kwadratowych jeszcze *przed* zastosowaniem zmniejszenia przez czynnik $\sum_{j=0}^{N(i)} C_i(j, x_k)^2 < 1$ (ostrą nierówność możemy wymusić przydzielając kompetencję więcej niż jednemu rozwiązaniu szczegółowemu). Obrazuje to możliwości zmniejszenia błędu, trudno się jednak spodziewać, by odchylenia aproksymacji miały dokładnie podany rozkład. W kolejnych przykładach oszacowanie dotyczy jeszcze bardziej specyficznych, lecz wciąż realistycznych warunków.

Przykład 2. Tylko dwa węzły potomne (wliczając aproksymację w danym węźle) mają niezerową funkcję kompetencji. Rozkład odchyleń tych aproksymacji na danej współrzędnej tworzy dwuwymiarowy rozkład normalny. Przeciętna wartość błędów kwadratowych rozwiązań składowych wynosi 1 (jest to też wariancja składowych wspomnianego rozkładu odchyleń), a przeciętna wartość odchyleń jest równa 0. Wartości funkcji kompetencji są równe c_1 i $c_2 = 1 - c_1$. Na wykresie jedną ze zmiennych jest różnica pomiędzy c_1 i c_2 w zakresie 0 do 0.98, drugą korelacja odchyleń w zakresie (-0.9 do 0.9). Na rysunku 3.1 znajduje się wykres błędu w zależności od różnicy i korelacji.

Zgodnie z uwagami podanymi powyżej, minimalne wartości błędów osiągane są przy równym ważeniu rozwiązań składowych, zwraca uwagę bardzo silna, niemal liniowa zależność od korelacji pomiędzy błędami.

Przykład 3. W tym przykładzie występują dwie różnice w stosunku do poprzedniego: oczekiwane błędy kwadratowe (będące jednocześnie wariancjami rozkładu odchyleń) są nierówne: większa wynosi 1.5, mniejsza 1. Ponadto większą wagę przydzielamy zawsze rozwiązaniu, które ma mniejszy błąd oczekiwany. Wykres błędu w zależności od różnicy i korelacji zamieszczono na rysunku 3.2.



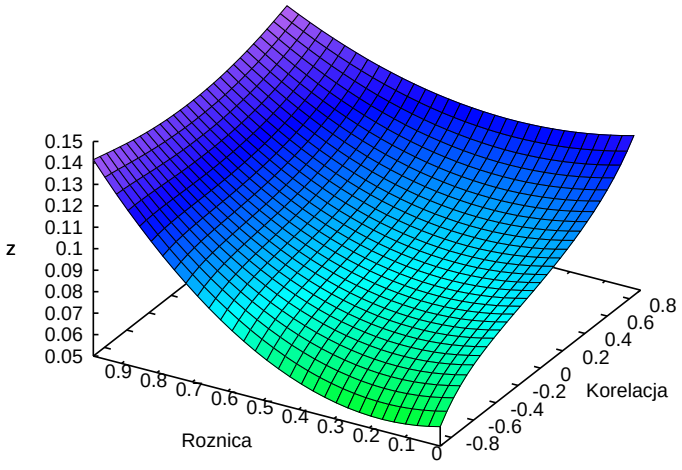
Rysunek 3.1. Wykres oczekiwanego błędu kwadratowego rysowany względem różnicy w przydzielonych funkcjach kompetencji oraz korelacji odchyleń. Dla równych wariancji odchyleń.

W tym przypadku minimum nie będzie już w zerze różnicy. Okazuje się, że przy podobnych założeniach da się osiągnąć wynik bardziej ogólny.

3.3.1. Wiedza o oczekiwanych błędach i o kowariancji odchyleń

Poniższy lemat dotyczy strategii przy znanym oczekiwanym odchyleniu i znanej wariancji odchylenia dla dwóch rozwiązań składowych.

Lemat 3. *Jeżeli wybrane zostały dwa rozwiązania składowe o pozycjach j_1 i j_2, których odchylenie dla danej współrzędnej wyjścia ma łącznie dwuwymiarowy rozkład normalny, jego wartość oczekiwana wynosi $[\mu_1, \mu_2]$, a macierz kowariancji V (ich wariancje V_{11} i V_{22}, a kowariancja $V_{12} = V_{21}$), ponadto $V_{12}^2 < V_{11} \cdot V_{22}$ oraz $V_{12}^2 +$*



Rysunek 3.2. Wykres oczekiwanego błędu kwadratowego rysowany względem różnicy w przydzielonych funkcjach kompetencji oraz korelacji odchyleń. Dla stosunku odchyleń równego 1.5.

$\mu_1 \cdot \mu_2 < V_{11} + \mu_1^2$, $V_{12}^2 + \mu_1 \cdot \mu_2 < V_{22} + \mu_2^2$, to najmniejszy błąd kwadratowy średniej ważonej (dla tej współrzędnej) występuje dla

$$\frac{C_i(j_2, x)}{C_i(j_1, x)} = \frac{V_{11} + {\mu_1}^2 - V_{12} - \mu_1\,\mu_2}{V_{22} + {\mu_2}^2 - V_{12} - \mu_1\,\mu_2} \tag{3.12}$$

$$= \frac{\mathbf{E}\left(\tilde{\eta}_{I(i,j_1)}(x)_l^2\right) - \mathbf{E}\left(\tilde{\eta}_{I(i,j_1)}(x)_l \cdot \tilde{\eta}_{I(i,j_2)}(x)_l\right)}{\mathbf{E}\left(\tilde{\eta}_{I(i,j_2)}(x)_l^2\right) - \mathbf{E}\left(\tilde{\eta}_{I(i,j_1)}(x)_l \cdot \tilde{\eta}_{I(i,j_2)}(x)_l\right)} \tag{3.13}$$

gdzie $\mathbf{E}$ *oznacza wartość oczekiwaną przy danym rozkładzie, z błędem kwadratowym*

$$\mathbf{E}(\tilde{\eta}_i(x)_l^2) = V_{hh} + {\mu_h}^2 - \frac{\left(V_{hh} - V_{12} - \mu_1\,\mu_2 + {\mu_h}^2\right)^2}{V_{22} + V_{11} - 2\,V_{12} + {\mu_2}^2 - 2\,\mu_1\,\mu_2 + {\mu_1}^2} \tag{3.14}$$

$$= \mathbf{E}\left(\tilde{\eta}_{I(i,j_h)}(x)_l^2\right) + - \frac{\left(\mathbf{E}\left(\tilde{\eta}_{I(i,j_h)}(x)_l^2\right) - \mathbf{E}\left(\tilde{\eta}_{I(i,j_1)}(x)_l \cdot \tilde{\eta}_{I(i,j_2)}(x)_l\right)\right)^2}{\mathbf{E}\left(\tilde{\eta}_{I(i,j_2)}(x)_l^2\right) + \mathbf{E}\left(\tilde{\eta}_{I(i,j_1)}(x)_l^2\right) - 2 \cdot \mathbf{E}\left(\tilde{\eta}_{I(i,j_1)}(x)_l \cdot \tilde{\eta}_{I(i,j_2)}(x)_l\right)}, \tag{3.15}$$

przy czym za h można podstawić zarówno 1, jak i 2 (wzór jest w tym względzie symetryczny).

Uzasadnienie jest przedstawione w dodatku A. Zatem w podanych warunkach

wagi powinny być odwrotnie proporcjonalne do oczekiwanych błędów kwadratowych, jednak różnica pomiędzy nimi powinna być zwiększona lub zmniejszona w zależności od wartości oczekiwanej iloczynu błędów. Zwiększenie wymagane jest przy dodatniej jego wartości, zmniejszenie przy ujemnej. W granicy, której osiągnięcie jest zabronione przez założenia, przy dużej wartości oczekiwanej iloczynu iloraz wartości funkcji kompetencji zmierza do nieskończoności lub do 0, co oznacza przydzielenie całej wagi rozwiązaniu składowemu z mniejszym błędem (w całkowicie zdegenerowanym przypadku do symbolu nieoznaczonego, w którym przydział kompetencji nie ma znaczenia).

Założenia tego twierdzenia są dosyć silne, dlatego trudno byłoby je wprost zastosować do realnej sytuacji. Przede wszystkim nie znamy wartości oczekiwanych błędów, a aproksymować je możemy tylko na odpowiednio dużych próbach, podczas gdy najbardziej przydatne byłyby oszacowania dla małych grup. Mogą też one różnić się znacznie od rozkładu normalnego. Ponadto, założenia na V_{12} i μ_1, μ_2 ograniczają stosowalność twierdzenia do przypadku, w którym nie ma jednocześnie dużej różnicy w oczekiwanych błędach i dużej korelacji tych błędów. Dodatkowo, lemat zachodzi dla każdej współrzędnej. Skoro zaś wyniki dla różnych współrzędnych mogą być różne, to nie da się zastosować go bezpośrednio do wyliczenia funkcji kompetencji dla przykładów z większą liczbą współrzędnych wyjścia w obecnym rozwiązaniu, w którym dla każdej współrzędnej funkcja kompetencji jest taka sama. Co prawda, w rozpatrywanym zakresie problemów (rozdz. 5, 6) problemy z jedną współrzędną są dość częste, a problem wielowymiarowy można łatwo podzielić na problemy o jednej współrzędnej. Pomimo tak wielu zastrzeżeń co do pełnej praktycznej spełnialności założeń lematu 3, może on pomóc w zobrazowaniu struktury błędu metody i tworzeniu strategii przydziału zbiorów uczących (por. rozdz. 4).

Istotne jest, że w równaniu (3.15) od dowolnego oczekiwanego błędu kwadratowego dla danej współrzędnej odejmowany jest składnik, który przy danych założeniach jest dodatni, co oznacza, że błąd kwadratowy będzie mniejszy od błędu kwadratowego *dowolnego* z wybranych węzłów. Wielkość tej różnicy zależy od własności rozkładu, ale np. przy zerowych wartościach oczekiwanych odchyleń oraz zerowej kowariancji wynosi ona $\frac{V_h h^2}{V_{22} + V_{11}}$, co przy dodatkowym założeniu na równość wariancji wynosi $V_h h/2$ i oznacza zmniejszenie błędu o połowę. Natomiast przy równych wariancjach, zerowych odchyleniach i całkowitym ujemnych skorelowaniu $V_{12} = -V_{11} = -V_{22}$, oczekiwany błąd zmniejsza się do 0.

Dla konstrukcji funkcji kompetencji i tworzenia rozwiązań składowych w węźle bardzo istotne jest również przywołane już powyżej ujawnienie dość znacznego wpły-

wu korelacji między odchyleniami rozwiązań składowych na możliwości osiągnięcia poprawnego wyniku.

3.4. Błąd węzła na zbiorze

W niniejszym podrozdziale przedstawione są autorskie twierdzenia, które są istotnym wkładem w teoretyczną część niniejszej dysertacji. W przeciwieństwie do poprzednich twierdzeń, dotyczących pojedynczych instancji lub specyficznych rozkładów błędu, w tej części zaprezentujemy wynik dotyczący błędu MSE na zbiorze, w odniesieniu do MSE na zbiorach kompetencji użytych rozwiązań składowych. Jest to o tyle istotne że, co prawda, istnieją twierdzenia dotyczące maszyn zbiorczych (komitetów) [67], ale poniższe dotyczą sytuacji, w której zbiory kompetencji tylko częściowo się nakładają, a mimo to są w stanie użyć MSE właśnie na zbiorach kompetencji.

Twierdzenie 1. *Dla węzła i oraz zbioru przykładów $\mathbf{S}$,*

$$MSE(\tilde{g}_i, \mathbf{S}) = \tau + \sum_{j=0}^{N(i)} \frac{|\mathbf{S}_{I(i,j)}|}{n_i^{max} \cdot |\mathbf{S}|} MSE(\tilde{g}_{I(i,j)}, \mathbf{S}_{I(i,j)}) + \\ + \frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} (n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2 - \varrho \tag{3.16}$$

gdzie:

1. *n_{ik} jest liczbą bezpośrednich potomków węzła i użytych dla przykładu k.*
2. *n_i^{max} jest maksymalną liczbą bezpośrednich potomków węzła i użytych dla przykładów ze zbioru $\mathbf{S}$.*
3. *$\tau = \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \tau_{kl}$ jest różnicą pomiędzy błędem, który osiąga ta funkcja kompetencji, a błędem, który osiągnęłaby funkcja kompetencji $\overline{C}$, która*
   a) *dla każdego przykładu x używa tych samych węzłów, co dana funkcja kompetencji,*
   b) *wszystkim wybranym węzłom potomnym przypisuje zawsze wagę $\frac{1}{n_i^{max}}$ z wyjątkiem aproksymacji w danym węźle (czyli "węzła" 0), którego waga jest zwiększana o $\frac{n_i^{max} - n_{ik}}{n_i^{max}}$, aby utrzymać sumę funkcji kompetencji równą 1.*

4. ϱ to nieujemny składnik wynikający pośrednio z (3.6).

$$\varrho = \frac{1}{|\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \varrho_{kl} \tag{3.17}$$

$$\varrho_{kl} = \left(\sum_{C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \frac{1}{n_i^{max}} + \frac{n_i^{max} - n_{ik}}{n_i^{max}} \cdot \eta_i(x_k)_l^2 \right) \cdot \\ \cdot \sin^2 \left(\left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)}, \left[\sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)} \right) \tag{3.18}$$

Przy czym

$$\sum_{j=0}^{N(i)} \frac{|\mathbf{S}_{I(i,j)}|}{n_i^{max} \cdot |\mathbf{S}|} + \frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} (n_i^{max} - n_{ik}) = 1 \tag{3.19}$$

Dowód znajduje się w dodatku A.

Wynik opisany w [21] był podobny, ale mniej ogólny.

Motywem przewodnim tego twierdzenia było wyodrębnienie z błędu średniokwadratowego całości wpływu błędów średniokwadratowych poszczególnych rozwiązań składowych. Podstawowym wyodrębnionym wpływem jest średnia ważona rozmiarami zbiorów kompetencji. Kolejny ze składników to błąd średniokwadratowy aproksymacji w węźle i, w którym błędy na niektórych przykładach mają wagę będącą całkowitą wielokrotnością wagi podstawowej (zob. pkt. 3 twierdzenia). Aproksymacja w węźle i została wybrana do tego celu dlatego, że jego zbiór kompetencji zawiera wszystkie przykłady, które zawierają zbiory kompetencji węzłów potomnych. Nie ma zatem niebezpieczeństwa, że błędy będą liczone na takich kombinacjach węzłów i przykładów, jakie nie powinny się zdarzyć w praktyce (czyli dane rozwiązania składowe w szczególności nie były uczone dla tych przykładów i mogłyby dawać większe błędy).

Należy zauważyć, że jeżeli dla każdego przykładu $n_i^{max} = n_{ik}$, to wszystkie składniki $n_i^{max} - n_{ik}$ wyniosą 0 i wyraz je zawierający może zostać wyeliminowany, co pozostawi tylko τ, ϱ i średnią ważoną błędów średniokwadratowych.

Równanie (3.16) poza składnikiem τ jest w istocie równaniem błędu dla funkcji kompetencji $\overline{C}$. Możliwe jest stosunkowo proste utworzenie takiej funkcji kompetencji na danym zbiorze na podstawie już istniejącej funkcji, zgodnie z pkt. 3 twierdzenia. Składnik τ miałby wtedy zagwarantowaną wartość 0. W praktyce nie jest to stosowane, gdyż składnik τ może być ujemny. W kontekście tego twierdzenia składnik τ można traktować jako miarę względnej jakości przydziału wartości funkcji kompetencji.

Dowód twierdzenia pozwala też uzyskać bezpośredni wzór na τ:

$$\begin{aligned}\tau_{kl} = \sum_{C_i(j,x_k)>0} \Bigg(& \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot \\ & \cdot \Bigg(C_i(j,x_k) \cdot \cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)}, \left[\sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)} \right) + \\ & - \frac{1}{n_i^{max}} \cdot \cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)}, \left[\sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)} \right) \Bigg) \Bigg) + \\ & - \frac{n_i^{max} - n_{ik}}{n_i^{max}} \cdot \eta_i(x_k)_l^2 \cdot \\ & \cdot \cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)}, \left[\sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)} \right) \\ \tau = & \frac{1}{|\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \tau_{kl}\end{aligned}$$

Pozostały składnik ϱ jest nieujemny, więc nie zwiększa błędu. Co więcej, wynika on z równania (3.6) dla zmodyfikowanej w opisany sposób funkcji kompetencji, stąd też wniosek 2 może zostać zastosowany analogicznie także do tej funkcji: $\varrho_{kl} = 0$ tylko wtedy, kiedy odchylenia aproksymacji są równe. Aby ich suma po wszystkich przykładach była równa 0, równość musiałaby zachodzić dla wszystkich przykładów. Stąd ϱ jest równe zero tylko gdy odchylenia wszystkich wybranych aproksymatorów są równe dla wszystkich przykładów i wszystkich współrzędnych wyjścia. Za wyjątkiem takiego, trudnego do uzyskania, przypadku błąd zostanie przez ten składnik zmniejszony.

3.5. Dodawanie poddrzewa a błąd średniokwadratowy

Na podstawie twierdzenia 1 możemy oszacować jak zmienia się błąd węzła, gdy dodajemy do niego poddrzewo w czasie uczenia. Błąd ten na zbiorze $\mathbf{S}$ zmniejszy się, o ile $\mathrm{MSE}(\tilde{g}_i, \mathbf{S}) < \mathrm{MSE}(g_i, \mathbf{S})$, co po użyciu równania (3.16) daje:

$$\begin{aligned}\mathrm{MSE}(g_i, \mathbf{S}) > & \\ & \tau + \sum_{j=0}^{N(i)} \frac{|\mathbf{S}_{I(i,j)}|}{n_i^{max} \cdot |\mathbf{S}|} \mathrm{MSE}(\tilde{g}_{I(i,j)}, \mathbf{S}_{I(i,j)}) + \\ & + \frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} (n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2 - \varrho \,. \end{aligned} \tag{3.20}$$

W szczególności będzie to prawdziwe gdy następujące warunki będą równocześnie spełnione:

1. Błędy węzłów potomnych *na ich zbiorach kompetencji* będą nie większe niż ich rodzica, na tych samych zbiorach.
2. Współczynnik względnej jakości $\tau \leqslant 0$, a więc rozwiązanie z daną funkcją kompetencji będzie dawało nie większy błąd niż wersja poddana modyfikacji z twierdzenia 1.
3. Istnieje przynajmniej jeden taki przykład x_k, dla którego $n_{ik} > 1$ i odchylenia wybranych do oceny przykładu aproksymatorów różnią się, co oznacza, że $\varrho > 0$.

Nie są to założenia szczególnie silne. Choć zagwarantowanie założenia 1 na nieznanych zbiorach może być trudne, to jednak nie jest ono całkiem bez szans na spełnienie, a na pewno są one większe niż np. wymaganie, żeby węzły te miały błąd mniejszy od swojego przodka na każdym przykładzie. Założenie 2 można ostatecznie wymusić ustalając funkcję kompetencji na $\overline{C}$, natomiast 3 jest spełnione prawie zawsze, z wyjątkiem bardzo szczególnego przypadku (por. koniec podrozdziału 3.4). Wymaganie $n_{ik} > 1$ będzie natomiast istotne w dalszych rozważaniach.

Oczywiście, to tylko przykładowe warunki i ogólna postać nierówności (3.20) pozwala, by niektóre nie były spełnione, o ile błąd wynikły z tego niespełnienia zostanie zniwelowany przez pozostałe. Są one jednak na tyle ogólne, że mogą zostać wykorzystane jako wskazówki do konstrukcji rozwiązań szczegółowych.

Jak wpłynie utworzenie nowego poddrzewa w węźle na całość drzewa? Jeżeli węzłem, któremu dodamy potomków, jest korzeń, spowoduje ono zmniejszenie MSE rozwiązania na danym zbiorze (spełniającym ww. warunki). Jeżeli węzłem, w którym dodajemy poddrzewo, nie jest korzeń, spowoduje ono, dla pewnych i i j, zmniejszenie MSE $(\tilde{g}_{I(i,j)}, \mathbf{S})$ w odpowiednim miejscu wzoru na błąd rodzica oraz, z pewnymi wagami, jego przodków i w ten sposób zmniejszy błąd drzewa. Może też spowodować zmiany ϱ, dla wszystkich przodków węzła w którym dodajemy poddrzewo. Zmiany te mogą błąd całości rozwiązania zarówno podnosić, jak i obniżać. Ze względu na te zmiany, trudno podać dokładne wyliczenie wpływu dodania poddrzewa w węźle innym niż korzeń.

3.6. Podsumowanie

Zamieszczone powyżej wyniki teoretyczne dotyczyły ogólnej struktury proponowanego rozwiązania niezależnej od uszczegółowień schematu uczenia (zob. podrozdział 2.4) wymaganych w jego punktach 2 (tworzenie aproksymatorów funkcji

w węzłach) i 6a, 6b (podział zbiorów uczących i funkcja kompetencji). Miały one dwa podstawowe cele. Po pierwsze wstępne uzasadnienie celowości zastosowanego ogólnego podejścia. Po drugie uzyskanie wskazówek do tworzenia wspomnianych uszczegółowień.

Najbardziej do pierwszego z tych celów przybliżają wnioski 1 i 2 i twierdzenie 1 z podrozdziału 3.5. Wnioski 1 i 2 mówią, że błąd na przykładzie jest zwykle mniejszy niż średnia ważona błędów kwadratowych użytych aproksymatorów (o ile użyty jest więcej niż jeden). Twierdzenie 1 natomiast uzasadnia podobne (choć nieco bardziej skomplikowane) stwierdzenie wobec błędu średniokwadratowego na zbiorze oraz średniej ważonej błędów średniokwadratowych aproksymatorów na ich zbiorach kompetencji w tym zbiorze. Ważną zaletą wspomnianych twierdzeń są ich bardzo ogólne założenia ograniczające się do stwierdzenia, że mamy do czynienia z danym rozwiązaniem, nie wszystkie węzły mają identyczne odchylenia na zbiorze, którym się interesujemy a aproksymator w rozważanym węźle ma przypisaną niezerową wartość funkcji kompetencji. Kolejną jest użycie zwykłych (nieważonych) błędów średniokwadratowych tylko na zbiorach kompetencji potomków co, o ile autorowi wiadomo, nie ma odpowiednika w literaturze dotyczącej rozwiązań wymienionych w podrozdziale 1.4. W zamian wyniki wspomnianych twierdzeń i obserwacji zawierają dość skomplikowane, choć dające się oszacować, zmienne.

W tym zakresie te ogólne twierdzenia wspomagane są przez obserwacje i przykłady dotyczące szczególnych przypadków rozkładów odchyleń (np. normalnych), takie jak przykład 3 (s. 31) i wcześniejsze. Te twierdzenia i przykłady dostarczają dokładniejszych oszacowań na wielkość błędu w bardziej specyficznych warunkach, np. oczekiwany błąd mniejszy niż dowolnego z dwóch użytych aproksymatorów z podanym wzorem w zależności od szczegółów rozkładu odchyleń. Ceną większej dokładności są jednak silniejsze założenia, np. na rozkład odchyleń oszacowań aproksymatorów od wartości poprawnej.

Drugi cel został spełniony przez obserwacje zawierające wzór na błąd na poszczególnych przykładach lub małych grupach, tzn. lemat 1 (s. 28) i wniosek 2, (s. 28) oraz, jako szczególny przypadek, przykład 3. Uzyskane wskazówki zostaną wykorzystane w następnym rozdziale.

4. Uszczegółowienia schematów algorytmów

Niniejszy rozdział przedstawia uszczegółowienia algorytmu uczenia podanego w podrozdziale 2.4 i algorytmu uzyskiwania wyniku z podrozdziału 2.3.

Rozwiązania podzadania podziału zbiorów uczących i tworzenia funkcji kompetencji przedstawione w tym rozdziale opierają się na niektórych wnioskach wynikających z twierdzeń z rozdziału 3.

1. Z lematu 1 oraz kolejnych wynika, że dla poszczególnych współrzędnych wektor odchyleń aproksymatorów i wektor funkcji kompetencji powinny tworzyć kąt jak najbliższy prostemu, a jak najdalszy zerowemu. Nie można tego zagwarantować bezpośrednio ponieważ, na interesujących nas nieznanych przykładach, nie znamy wektora odchyleń. Można jednak zastosować pośrednie wersje tej techniki.
   Z równania (3.6), w którym występuje średnia ważona, bezpośrednio wynika, że gdyby nie dało się nic powiedzieć o kącie odpowiednich wektorów, to najlepiej wybrać aproksymację z minimalnym oczekiwanym błędem. Drugi czynnik (kwadrat cosinusa) może być jednak mniejszy od 1, a nawet bliski 0, tylko jeżeli wybrany został więcej niż jeden węzeł.
   Z kolei obserwacja z lematu 3 dla dwuwymiarowego normalnego rozkładu odchyleń dwóch aproksymatorów sugeruje funkcję kompetencji odwrotnie proporcjonalną do oczekiwanego błędu kwadratowego, z dokładnością do dodatkowych składników wynikających z korelacji odchyleń.
   Obydwa te wyniki mają wspólny bardzo ogólny aspekt: funkcja kompetencji przyjmuje większą wartość dla aproksymacji ze spodziewanym mniejszym błędem (stąd zresztą nazwa "funkcja kompetencji"). Nie należy jednak ograniczać zanadto w ten sposób liczby wybranych aproksymatorów. Jako że oszacowanie błędu jest w dużej mierze zgrubne i bardzo pośrednie, dokładna formuła ma mniejsze znaczenie, ale podstawą jest odwrotna proporcjonalność, być może z progami odcięcia, aby nie używać aproksymatorów o szczególnie dużym oczekiwanym błędzie.
2. Wspomniane wyżej obserwacje oraz twierdzenie 1 unaoczniają potrzebę tworzenia zbiorów uczących oraz funkcji kompetencji tak, aby jeden przykład był

oceniany przez więcej niż jeden aproksymator, a ponadto odchylenia aproksymatorów dla tego przykładu różniły się między sobą – najlepiej miały przeciwne znaki. Jest to potrzebne, by czynnik z cosinusem w równaniu (3.10) mógł być zerowy – skoro wszystkie współrzędne wektora kompetencji są nieujemne, więc zerowy cosinus nie zostanie osiągnięty dla żadnego wektora ze współrzędnymi jednego znaku. Odpowiada to wystąpieniu niskiej, najlepiej ujemnej wartości oczekiwanej iloczynu odchyleń w lemacie 3, co obniżałoby wartość oczekiwaną błędu w umieszczonym tam wzorze.

Podstawowym sposobem, który można zastosować przy tworzeniu zbiorów uczących w celu realizacji wspomnianej potrzeby, jest przydzielanie różnym aproksymacjom różniących się, ale częściowo pokrywających (by umożliwić pokrywanie zbiorów kompetencji i ocenę jednego przykładu przez więcej aproksymatorów) zbiorów uczących.

3. Oczywiście, błąd całości rozwiązania będzie mniejszy, jeśli błędy kwadratowe aproksymatorów składowych będą mniejsze. Zgodnie z twierdzeniem 1 oraz wynikami opisanymi w podrozdziale 3.5, w przypadku tworzenia poddrzewa w danym węźle dobrze byłoby uzyskać błędy aproksymatorów potomnych mniejsze niż miał ich rodzic na ich zbiorach kompetencji.

   Dla celów tworzenia funkcji kompetencji i zbiorów uczących przyjęte zostało następujące założenie: aproksymacje mają szansę uzyskać mniejsze błędy na swoich zbiorach kompetencji, o ile te zbiory oraz odpowiednie zbiory uczące będą w jakimś sensie prostsze od zbiorów kompetencji i uczącego rodzica.

Podstawowym rodzajem wspomnianego wyżej uproszczenia jest spowodowanie, żeby przykłady w danym zbiorze były do siebie "podobne". Konkretyzacje konceptu "podobieństwa" są zawarte w poszczególnych sposobach konstrukcji zbiorów kompetencji opisanych poniżej. Jedną z ich cech jest używanie odpowiedzi aproksymatora rodzica jako przynajmniej jednej z podstaw do wyznaczania "podobieństwa". Związane jest to głównie z założeniem mówiącym, że przykłady o podobnych wartościach funkcji wyjściowej rodzica mogą być do siebie "podobne" dla aproksymatorów węzłów potomnych. Jeśli zatem ograniczymy zakres wartości wyjścia, których nauczyć ma się aproksymator, jego zadanie może stać się łatwiejsze. Można na to liczyć zwłaszcza jeżeli aproksymatory na różnych poziomach mają podobną konstrukcję. Oczywiście wspomniane przykłady mogą mieć bardzo różne cechy wejściowe, dlatego one też mogą zostać użyte. Utożsamienie "podobieństwa" przykładów z niewielką różnicą w prawdziwej wartości szukanej funkcji dla tych przykładów bardzo ułatwiłoby rozważania, jednak takie dane nie są dostępne w czasie działania programu, więc mogą pełnić jedynie funkcję pomocniczą.

Nieco bardziej zaskakujące może być użycie miary podobieństwa w punkcie 1. W metodzie opisanej w podrozdziale 4.1, podobieństwo nieznanego przykładu do centrum zbioru uczącego danego węzła jest traktowane jako bardzo niedokładna miara spodziewanego błędu tego aproksymatora na tym przykładzie – im bardziej dany przykład jest "podobny" do przykładów ze zbioru uczącego danego węzła, tym mniejszego spodziewamy się błędu. Należy tu zauważyć, że gdyby używana była regresja liniowa, takie oszacowanie byłoby niepoprawne, jako że w takim przypadku przykłady bliżej brzegów zbioru uczącego mają większą wagę przy określaniu współczynnika regresji niż te bliżej środka przedziału regresji ([87]).

Wspomniane powyżej metody realizacji wskazówek wynikłych z twierdzeń można podsumować następująco: przykłady z jednego zbioru uczącego i kompetencji powinny być do siebie podobne. Zbiory uczące różnych aproksymatorów powinny być różne, choć zbiory kompetencji muszą mieć niepuste przecięcie, co ogranicza różnice w zbiorach uczących. Funkcja kompetencji ma być większa dla przykładów bardziej "podobnych" do zbioru uczącego. Powinna ona również być niezerowa dla większej liczby węzłów dla danego przykładu.

Są to własności bardzo podobne do własności wymaganych od wyniku klastrowania rozmytego – przykłady w jednym klastrze są podobne, w różnych różne, ale klastry mają niepuste przecięcie. Stąd taki rodzaj grupowania jest używany we wszystkich wersjach tworzenia funkcji kompetencji i zbiorów uczących węzłów potomnych.

Klastrowania w przestrzeni wyjściowej nie można nazwać metodą typową dla regresji w statystyce, choć np. podział dziedziny stosowany jest w technikach typu *spline*. Jest ono natomiast powiązane z metodami rozmytymi, a konkretnie z rozmywaniem zmiennej wyjściowej, które z kolei powoduje przesunięcie zadania w stronę klasyfikacji. Wydaje się to dość naturalne, gdyż przydział przykładów do zbiorów kompetencji węzłów potomnych jest pewnego rodzaju klasyfikacją. Trzeba jednak pamiętać, że jest to zastosowanie pomocnicze. Podstawą wciąż pozostaje dopasowanie funkcji o przeciwdziedzinie będącej przestrzenią wektorów o składowych rzeczywistych, przeprowadzane w każdym węźle. Z tego powodu rozwiązanie można zaliczyć do metod hybrydowych, w szczególności do metod neuronalno-rozmytych w razie zastosowania sieci neuronowych jako aproksymatorów w węzłach (co jest zwykle czynione).

Dodatkowe warunki, które miały wpływ na generowanie algorytmu tworzenia funkcji kompetencji i podziału zbioru uczącego, dotyczyły generalizacji, efektywności oraz liczby węzłów potomnych. Zasadniczo, przy większej liczbie wybranych węzłów dla danego przykładu, można się spodziewać większych zysków w stosunku do śred-

niego błędu, jednak włączanie przykładów do zbioru kompetencji wielu aproksymatorów prowadzi również do powiększenia sumarycznej wielkości zbiorów uczących węzłów potomnych oraz może prowadzić do zwiększenia błędów indywidualnych. Metody przedstawione poniżej stanowią kompromis pomiędzy tymi wymaganiami. Ostatecznym kryterium oceny są wyniki eksperymentalne (rozdz. 5 i 6).

Przykład najprostszej funkcji

W świetle powyższych rozważań najprostszą funkcją kompetencji oraz funkcją charakterystyczną zbiorów uczących mogłaby być funkcja przynależności z klastrowania rozmytego wykonanego na wynikach aproksymacji w danym węźle. W przypadku użycia jednego z najpopularniejszych algorytmów grupujących, rozmytego c-średnich [72], dodatkowo trzeba znaleźć liczbę klastrów (np. [22, 92]), a ponadto zastosować progowanie (np. takie jak opisane w podrozdziale 4.3.2). Progowanie jest konieczne, gdyż w innym przypadku zbiorami uczącymi i kompetencji byłby zwykle cały zbiór rodzica. Dzieje się tak dlatego, ponieważ funkcja przynależności tego algorytmu klastrowania przybiera niezerowe wartości dla wszystkich klastrów dla każdego przykładu, o ile przykład ten nie jest identyczny z centroidem jednego z klastrów.

Taka funkcja dobrze ilustruje podstawowe koncepcje tworzenia funkcji kompetencji i przydziału zbiorów uczących, ale nawet wstępne testy wykazują, że w takiej formie daje większe błędy niż metoda przedstawiona poniżej. Ta prosta metoda jest jednak podstawą bardziej zaawansowanych wariantów, w tym tego opisanego poniżej.

4.1. Metoda podziału zbioru uczącego i tworzenia funkcji kompetencji

W tej metodzie w jednym zbiorze uczącym oraz w jednym obszarze kompetencji umieszczane są przykłady podobne ze względu na odpowiedzi rodzica lub ze względu na zmienne wejściowe. Większa wartość funkcji kompetencji przyznawana jest tym węzłom, do zbiorów uczących których przykład jest bardziej podobny.

Algorytm 3. 1. Odpowiedzi rodzica są łączone z wektorami wejściowymi dla każdego przykładu. Na rezultacie tej operacji używany jest algorytm analizy składowych głównych (PCA [48]) i tworzona jest macierz złożona ze znormalizowanych wektorów własnych macierzy kowariancji. Wybierane są do niej wektory odpowiadające największym wartościom własnym, w najmniejszej liczbie

takiej, żeby suma wartości bezwzględnych wartości własnych odpowiadających tym wektorom była większa od 90% sumy wartości bezwzględnych wszystkich wartości własnych macierzy kowariancji. Przyjęto 12 jako dodatkowe ograniczenie na liczbę wektorów.

2. Dane są przekształcane z użyciem tej macierzy (mnożone). Następnie wywoływany jest algorytm klasteryzacji - ważone rozmyte c-średnich [72] (także podrozdział 4.3.1). Waga przykładu odpowiada pierwiastkowi z błędu kwadratowego aproksymatora rodzica na tym przykładzie. W tym miejscu sprawdzane jest rozłożenie klastrów. Jeżeli centroidy są zbyt blisko siebie, liczba wymiarów wyjściowych jest ograniczana do 4. Do znajdowania właściwej liczby klastrów można użyć algorytmu z pracy [22], jednak wstępne eksperymenty wskazały, że użycie stałej liczby, np. 5, jak w przypadku zamieszczonych w tej pracy eksperymentów, zwykle daje dobre rezultaty, a jest mniej kosztowne obliczeniowo.
3. Wartości przynależności otrzymane w poprzednim punkcie są częściowo progowane zgodnie z algorytmem opisanym w podrozdziale 4.3.2, tak aby przykłady miały niezerowe funkcje przynależności do "średnio w przybliżeniu" 3 klastrów.
4. Sprogowana funkcja przynależności zostaje funkcją przydziału do zbiorów uczących.
5. Sprogowana funkcja przynależności zostaje funkcją kompetencji, z tym że pewna część "kompetencji", mianowicie $\frac{1}{4(N(i)-1)}$, jest zawsze przyznawana aproksymacji w danym węźle. Następnie wartość przynależności jest renormalizowana.

Ograniczenie wymiarowości przestrzeni jest spowodowane tendencją algorytmu rozmytych c-średnich do grupowania wszystkich centroidów w centrum zbioru przy dużej liczbie wymiarów [98]. We wstępnych testach zjawisko to występowało prawie zawsze przy liczbie wymiarów powyżej 12. Niekiedy, w zależności od konkretnych danych i punktów startowych algorytmu (zob. punkt 4.3.1) występowało także dla niższych liczb wymiarów, lecz nigdy dla mniejszych niż 5. Stąd procedura ograniczenia wymiarowości opisana w algorytmie.

Przeciętna liczba niezerowych funkcji przynależności również została wybrana na podstawie wstępnych eksperymentów.

4.1.1. Warianty

Dla celów porównawczych wykorzystano dodatkowy wariant powyższej funkcji kompetencji nie używający wag – wszystkie przykłady są ważone jednakowo.

W trakcie powstawania tego rozwiązania opracowano kilka metod, spośród których dość istotnymi były metody "podwójnego klastrowania" i klastrowania macie-

rzy "pomyłek", opisane w pracach autora niniejszej dysertacji [20, 21, 24]. Pierwsza opierała się na grupowaniu rozmytym w prostym iloczynie wyjść aproksymatora rodzica i poprawnych odpowiedzi, przy czym funkcja kompetencji wykorzystywała specyfikę metod klastrowania używających centroidów, aby otrzymać funkcję kompetencji, która nie może zależeć od prawdziwych wartości szukanej funkcji. Druga polegała na wcześniejszej "granulacji" zarówno wyjść aproksymatora rodzica jak i prawdziwych wartości szukanej funkcji i tworzyła z niej macierz korelacji przynależności do danych klastrów na podstawie której tworzono odpowiednie funkcje. Opisywana w niniejszej pracy metoda uzyskała w testach niższe błędy od proponowanych wcześniej.

4.2. Tworzenie aproksymatorów składowych

Aproksymatory składowe, umieszczone w węzłach drzewa tworzonego przez metodę opisaną w niniejszej pracy mogą być, co do zasady, dowolnego typu. W niniejszej pracy zostały użyte sieci neuronowe typu feed-forward z jedną warstwą neuronów ukrytych oraz logistyczną funkcją aktywacji neuronów. Dla neuronów ukrytych jest ona pomnożona przez 2 i pomniejszona o 1 dla uzyskania zakresu [-1,1] i oczekiwanej wartości aktywacji 0 (jak sugerowano w pracy [60]). Liczba neuronów ukrytych jest w sposób bardzo przybliżony ustalona, za pracą [76]:

$$\left\lceil min\left(\left\lceil\sqrt{p+r^2}\right\rceil, \frac{\frac{|\mathbf{U}_i|}{\vartheta} - r}{(p+r+1)}\right)\right\rceil \tag{4.1}$$

gdzie p to wymiar przestrzeni cech, r - wymiar przestrzeni wyjściowej, $\mathbf{U}_i$ zbiór uczący w węźle i, a ϑ to parametr, który odpowiada założeniu, że liczba przykładów uczących powinna być kilkakrotnie (np. pięciokrotnie, stąd w niniejszej pracy $\vartheta = 5$) większa od liczby parametrów - połączeń. Taka sieć neuronowa jest na tyle mała, że pozwala mieć nadzieję na uniknięcie nadmiernego dopasowania i względnie krótki czas uczenia. Sieć nauczana jest metodą Levenberga-Marquardta [97] (krótko opisaną w podrozdziale 4.3.3). Jeżeli zbiór uczący danego węzła zawiera zbyt mało przykładów, aby nawet sieć o jednym neuronie ukrytym mogła otrzymać trzy razy więcej przykładów uczących, niż ma wag, sieć nie jest tworzona, chyba że węzłem, w którym mamy tworzyć sieć, jest korzeń – jednak w takim wypadku używana jest regresja liniowa. W przypadku, gdy zgodnie z punktem 4 algorytmu uczenia (str. 22), sieć jest uczona ponownie, liczba neuronów ukrytych zwiększana jest o 1, o ile

w takim przypadku zostanie zachowany warunek mówiący o odpowiednim ilorazie liczby przykładów do liczby wag. Takie dodatkowe uczenie jest wykonywane co najwyżej jednokrotnie.

4.3. Algorytmy pomocnicze

W niniejszym podrozdziale opisano pokrótce algorytmy pomocnicze, niektóre istniejące w literaturze, niektóre wykonane dla opisywanego rozwiązania. Metoda walidacji wyników rozmytych c-średnich, przydatna przy znajdowaniu właściwej liczby klastrów, opisana w pracy autora [22], może być wykorzystana w metodzie podziału zbioru kompetencji i przydziału funkcji przynależności opisanej powyżej, jednak ostatecznie nie została użyta w eksperymentach, a więc nie jest tu opisana.

4.3.1. Rozmyte c-średnich

Rozmyte c-średnich [15, 56, 72] jest jednym z najbardziej znanych algorytmów klastrowania rozmytego. Jego zaletami, dzięki którym znalazł zastosowanie w niniejszej pracy, są prostota, znane właściwości oraz akceptowalna złożoność czasowa, liniowa ze względu na liczbę przykładów pomnożoną przez liczbę klastrów, wymiarów i iteracji.

Funkcja celu, którą minimalizuje rozmyte c-średnich, ma postać [15, 72]:

$$J_m = \sum_{j=1}^{c} \sum_{i=1}^{|\mathbf{S}|} \mu_{ij}^m \; d(x_i, c_j)$$

gdzie x_i to i-ty wektor wejściowy, c_j to centroid j-tego klastra, d to pewna metryka, μ_{ij} to stopień przynależności i-tego wektora do j-tego klastra, c to liczba klastrów, a $|\mathbf{S}|$ to liczba wektorów. Przy tym każde $\mu_{ij} \in [0, 1]$ oraz $\sum_{j=1}^{c} \mu_{ij} = 1, \quad \forall i : x_i \in S$, co nazywane jest warunkiem klastrowania probabilistycznego [56]. W niniejszej pracy przyjęto wartość parametru $m = 2$.

Po inicjalizacji albo poprzez ustanowienie losowych przynależności, albo, jak w przypadku niniejszej pracy, poprzez ustanowienie losowych centroidów, następuje naprzemienna optymalizacja. W jednym kroku zmieniane są stopnie przynależności wg wzoru:

$$\mu_{ij} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(x_i, c_j)}{d(x_i, c_l)} \right)^{\frac{2}{m-1}}}$$

a w następnym centroidy:

$$c_j = \frac{\sum_{i=1}^{|\mathbf{S}|} \mu_{ij}^m \cdot x_i}{\sum_{i=1}^{|\mathbf{S}|} \mu_{ij}^m}$$

Dla celów, dla których jest używane w tej pracy, rozmyte c-średnich ma jednak dwie podstawowe wady: konieczność wyboru liczby klastrów oraz fakt, że przynależności do wszystkich klastrów są niezerowe dla wszystkich przykładów, które nie pokrywają się z centroidami klastrów. Liczbę klastrów ustawiono po wstępnych eksperymentach na jedną konkretną liczbę (5), co dało zadowalające rezultaty, choć dostępne są też algorytmy wyboru liczby klastrów [22, 72, 92]. Druga cecha wymagała użycia prostego algorytmu wspomnianego w następnym podrozdziale.

4.3.2. Progowanie wartości funkcji przynależności

Niedogodnością związaną z użyciem rozmytych c-średnich dla celów tej pracy są niezerowe wartości funkcji przynależności na całej przestrzeni dla wszystkich klastrów, z wyjątkiem przypadku identyczności przykładu z jednym z centroidów. Przy bezpośrednim zastosowaniu przekładałoby się to przede wszystkim na uczenie węzłów na dużych zbiorach uczących, tej samej wielkości co rodzica, co nie pomaga przy zmniejszaniu błędu, a zwiększa zapotrzebowanie na moc obliczeniową. Tak nauczone węzły byłyby używane także na przykładach, na których ich wkład w końcowy wynik byłby pomijalny. Stąd używany jest prosty algorytm, w którym wartości poniżej pewnego progu są uznawane za równe zeru, a całość jest renormalizowana do sumy równej 1.

Pewnym wyzwaniem może być tutaj wybór progu. W niniejszej pracy przyjęto, że korzystne jest, aby więcej niż jeden węzeł potomny miał niezerową funkcję kompetencji na danym przykładzie. Przybliżenie tego wyniku osiągnięte zostało w następujący sposób: ustalona jest pewna liczba a, we wszystkich opisanych eksperymentach równa 3. Dla każdego przykładu ze zbioru kompetencji obliczane są pierwotne wartości funkcji przynależności a następnie, wśród wartości funkcji przynależności uporządkowanych od największej, znajdowane są dwa elementy: $\overline{\mu}$ – o randze $\lceil a + 0.5 \rceil$ oraz $\underline{\mu}$ – o randze $\lfloor a + 0.5 \rfloor$ (w stosowanych rozwiązaniach ich istnienie jest zapewnione). Jako próg użyta jest średnia po wszystkich przykładach z wartości $\{a + 0.5\}\overline{\mu} + (1 - \{a + 0.5\})\underline{\mu}$. Jeżeli progowanie dla danego wektora i przykładu skutkowałoby wektorem zerowym, co może się zdarzyć przy małych wartościach a, przykładowi jest przypisywana przynależność 1 do klastra, dla którego stopień przynależności przed progowaniem był największy.

Powyższy algorytm jest autorski.

4.3.3. Metoda uczenia sieci neuronowych

W niniejszej pracy do uczenia sieci neuronowych używana jest metoda Levenberga-Marquardta [97] z regularyzacją. Oryginalna metoda minimalizuje sumę kwadratów błędów (lub ogólnie - wartości funkcji), natomiast użyta implementacja minimalizuje:

$$B(x) = \frac{1}{2} \sum_{i:x_i \in X} ||y_i - g(x_i)||^2 + \alpha ||w||^2 \tag{4.2}$$

gdzie X jest zbiorem uczącym, x_i i-tym wektorem wejściowym, y_i - wyjściowym, $g(x_i)$ odpowiedzą sieci neuronowej dla x_i, w wektorem wag połączeń sieci neuronowej, a $|| \cdot ||$ oznacza normę euklidesową. Parametr α został po wstępnych eksperymentach ustawiony na 0.001, nie były stosowane tutaj metody jego aktualizacji, takie jak np. opisana w [78]. Jego rolą było zapobieganie powstawaniu wag o bardzo dużych wartościach, które mogłyby negatywnie wpływać na generalizację. Można zauważyć, że ten parametr nie jest zależny od liczby wag i przykładów, przez to jego wpływ jest tym większy, im mniej jest przykładów, a więcej wag. Jest to właściwość pożądana – mając więcej przykładów w stosunku do wag, możemy mieć większą ufność co do ew. powstałych dużych wartości wag.

Metoda Levenberga-Marquardta w kroku k przeszukuje przestrzeń wag w kierunku δw_k wyznaczonym przez:

$$(J_k^T J + \lambda_k I)\delta w_k = -J_k^T e_k \tag{4.3}$$

gdzie J to jakobian błędu po wagach, a e_k to wektor błędów w danym kroku. Niektóre rozwiązania zastępują macierz identyczności przekątną $J_k^T J$.

W powyższym algorytmie λ_k jest na początku ustawione na wartość λ_0. W każdym kolejnym kroku, jeżeli poprzedni krok zmniejszył błąd, to $\lambda_k = \lambda_{k-1}/\lambda_-$, w przeciwnym wypadku $\lambda_k = \lambda_{k-1} \cdot \lambda_+$. W niniejszej pracy we wszystkich eksperymentach przyjęto λ_-=4, λ_+=2, $\lambda_0 = \frac{1}{16}$. Zastosowano także metodę tzw. wielostartu, dokładniej – losowano dwa zestawy wag początkowych, uczono sieci, a następnie wybierano tę o mniejszym błędzie na zbiorze uczącym. Warunek stopu we wszystkich eksperymentach ustalono na 100 epok albo średni błąd kwadratowy nie większy niż 10^{-5}.

4.4. Oszacowanie złożoności czasowej z użytymi rozwiązaniami szczegółowymi

Złożoność obliczeniowa metody proponowanej w niniejszej pracy zależy w istotny sposób od złożoności użytych algorytmów składowych i pomocniczych, dlatego została opisana dopiero w tym podrozdziale.

4.4.1. Liczba węzłów i rozmiary zbiorów uczących

W pełnym drzewie przy liczbie bezpośrednich potomków k i wysokości n jest $\frac{k^{n+1}-1}{k-1}$ węzłów. Gdyby przyjąć za k maksymalną liczbę dzieci dowolnego węzła, to wtedy liczba ta jest górnym oszacowaniem liczby węzłów w drzewie.

Drzewo aproksymatora hierarchicznego nie musi i często nie jest pełne z uwagi na warunek 3 lub 4 algorytmu uczenia (s. 22). Poza tym, w ogólnym przypadku, liczby dzieci węzłów mogą się różnić (choć w rozwiązaniu szczegółowym użytym w niniejszej pracy zrezygnowano zrezygnowano z tej możliwości (zob. podrozdział 4.1). Wspomniane oszacowanie dotyczy zatem maksymalnej liczby węzłów.

Jeśli suma liczności zbiorów uczących na każdym kolejnym poziomie jest a razy większa od sumy liczności zbiorów uczących na poprzednim poziomie, to suma liczności zbiorów uczących wynosi nie więcej niż $\frac{a^{n+1}-1}{a-1}|\mathbf{S}|$. To oszacowanie może być jednak znacząco zbyt wysokie.

Węzłów wewnętrznych jest odpowiednio maksymalnie $\frac{k^n-1}{k-1}$, gdzie k jest maksymalną liczbą dzieci danego węzła, a suma liczności ich zbiorów uczących wynosi $\frac{a^n-1}{a-1}|\mathbf{S}|$.

Podsumowując, ponieważ aproksymacja może być tworzona i uruchamiana w każdym węźle, a zbiory uczące tworzone są i kompetencje wyznaczane w węzłach wewnętrznych, zachodzi następujący lemat:

Lemat 4. *Jeżeli zbiór uczący ma rozmiar $|\mathbf{S}|$, suma liczności zbiorów uczących na każdym kolejnym poziomie jest nie więcej niż a razy większa od sumy liczności zbiorów uczących na poprzednim poziomie ($a > 1$), to*

1. *Suma rozmiarów zbiorów uczących w drzewie wynosi co najwyżej $\frac{a^{n+1}-1}{a-1}|\mathbf{S}|$.*
2. *Suma rozmiarów zbiorów uczących w węzłach wewnętrznych wynosi co najwyżej $\frac{a^n-1}{a-1}|\mathbf{S}|$.*

Analogicznie, w czasie uzyskiwania predykcji na zbiorze $|\mathbf{T}|$, przy stosunku sumy liczby węzłów o niezerowej kompetencji na kolejnym poziomie do liczby takich węzłów na poziomie poprzednim nie większym niż a

1. *Sumaryczna liczba przykładów poddanych pod ocenę aproksymatorów w węzłach wynosi co najwyżej* $\frac{a^{n+1}-1}{a-1}|\mathbf{T}|$.
2. *Sumaryczna liczba przykładów, dla których trzeba wyznaczyć kompetencję, wynosi co najwyżej* $\frac{a^{n}-1}{a-1}|\mathbf{T}|$.

4.4.2. Przykład złożoności dla podstawowej wersji - uczenie

W czasie uczenia w każdym węźle trzeba utworzyć i uczyć sieć neuronową, ponadto w każdym węźle wewnętrznym trzeba utworzyć węzły potomne.

Uczenie aproksymatora w węźle i składa się z:

1. Co najwyżej liniowej, $O(|\mathbf{S}_i|(r+p))$, obróbki wstępnej, w tym utworzenia sieci i inicjacji wag.
2. Uczenia sieci algorytmem Levenberga-Marquardta (LM).
3. Sprawdzenia błędów w węźle.

Ponieważ wag oraz progów sieci jest w sumie nie mniej niż neuronów, ale mniej niż przykładów, jedna iteracja algorytmu LM zajmuje $O(rw_u^2|\mathbf{S}_i|)$, gdzie w_u to liczba progów i wag w sieci użytych dla obliczenia jednej współrzędnej wyjścia dla jednego przykładu. Wynika to z konieczności wyliczenia quasi-hesjanu. Wzór ten uwzględnia wyliczanie pochodnych dla każdego przykładu i każdego wymiaru wyjściowego oddzielnie. Zoptymalizowana implementacja jest natomiast w stanie uwzględnić, że przy obliczaniu pochodnej jednego z wyjść wagi od neuronów ukrytych do pozostałych wyjść mają pochodną 0 i możemy je pominąć. Jedna iteracja algorytmu zajmuje zatem $O(rw_u^2|\mathbf{S}_i|it_{lm})$, gdzie it_{lm} jest maksymalną liczbą epok metody Levenberga-Marquardta. Zatem, jeśli w_u jest maksymalną liczbą użytych wag sieci, a it_{lm} maksymalną liczbą epok, dla wszystkich węzłów suma działania wynosi: $O\left(\frac{a^{n+1}-1}{a-1}|\mathbf{S}|rw_u^2 \cdot it_{lm}\right)$. W sprawdzeniu błędów dominuje operacja uzyskiwania wyników, która, przy założeniu, że w jest liczbą wszystkich wag sieci, ma złożoność $O(w|\mathbf{S}_i|)$, a więc jest tu pomijalna.

Jak widać, złożoność obliczeniowa metody Levenberga-Marquardta zależy przede wszystkim od liczby wag, dlatego też jest ona polecana dla niewielkich sieci. Założeniem proponowanego rozwiązania jest jednak łączenie prostych rozwiązań składowych, a więc używane są właśnie sieci niezbyt wielkie. Dokładniejszy rozmiar można wyznaczyć ze wzoru (4.1) podanego w podrozdziale 4.2. Dla uproszczenia załóżmy, że część ograniczająca liczbę neuronów ukrytych ze względu na liczbę przykładów nie została wykorzystana, czyli $\frac{\frac{|\mathbf{S}_i|}{\vartheta}-r}{(p+r+1)} > \left\lceil\sqrt{p+r^2}\right\rceil$. Skoro mamy $\left\lceil\sqrt{p+r^2}\right\rceil$

węzłów ukrytych, to mamy też

$$w = \left\lceil \sqrt{p + r^2} \right\rceil (p + r + 1) + r$$

wag i progów w sieci neuronowej. W czasie uczenia liczba wag użytych dla jednej współrzędnej wyjścia i jednego przykładu może być podana jako:

$$w_u = \left\lceil \sqrt{p + r^2} \right\rceil (p + 1 + 1) + 1$$

Dla uproszczenia zapiszmy powyższy wzór jako $O(\sqrt{p + r^2}(p + 1 + 1) + 1)$. Pamiętając też, że $O(a+b)^2 = O(a^2+b^2+2ab)$, a $\sqrt{p + r^2}(p+1+1) > r$ oraz $p+r \geqslant 2$, główny wzór przyjmie postać:

$$O\left(\frac{a^{n+1} - 1}{a - 1}|\mathbf{S}|r(\sqrt{p + r^2}(p + 2))^2 \cdot it_{lm}\right) = \tag{4.4}$$

$$= O\left(\frac{a^{n+1} - 1}{a - 1}|\mathbf{S}|r(p + r^2)(p + 2)^2 \cdot it_{lm}\right) \tag{4.5}$$

Tworzenie zbiorów uczących wymaga:

1. Uzyskania wyników węzła rodzica (co jest już robione przy uzyskiwaniu jego błędów).
2. Obróbki wstępnej o złożoności $O(|\mathbf{S_i}|(r + p)^2)$ (obliczanie macierzy kowariancji).
3. Analizy składowych głównych (PCA) ($O((r + p)^3)$).
4. Przekształcenia przestrzeni przez transformację znalezioną przez PCA - $O(|\mathbf{S_i}|(r+p)^2)$ (mniej, jeśli PCA odrzuciła jakieś współrzędne).
5. Grupowania przez rozmyte c-średnich, które w tym przypadku ma złożoność $O(k|\mathbf{S_i}|(r + p)it_{fcm})$, gdzie it_{fcm} to liczba iteracji rozmytego c-średnich.
6. Uzyskania stopni przynależności przykładów do klastrów otrzymanych za pomocą rozmytego c-średnich (co jednak może być robione razem z grupowaniem).
7. Progowania, wraz ze znalezieniem progu – $O(k \log k|\mathbf{S_i}|)$.
8. Obróbki końcowej (przydziału przykładów do zbiorów) wymagające $O(a|\mathbf{S_i}|)$ operacji, jeżeli same przykłady nie są w żaden sposób modyfikowane.

Skoro $|\mathbf{S_i}| > (r + p), it_{fcm} > k$, można oszacować czas potrzebny na utworzenie funkcji kompetencji i przydział zbiorów uczących w danym węźle jako $O(|\mathbf{S_i}|(k(r + p)it_{fcm} + (r + p)^2 + a))$. Zatem ta część zajmuje znacząco mniej czasu niż uczenie aproksymatorów, zwłaszcza że jest wykonywana tylko w węzłach wewnętrznych.

Podsumowując: uczenie aproksymatorów jest w uczeniu wersji rozwiązania opisanej w dysertacji operacją dominującą. Czas zużyty na nią jest zależny wykładniczo

od wysokości drzewa – por. wzór (4.4). Podnoszony do potęgi jest stosunek sumy rozmiarów zbiorów uczących węzłów z kolejnego poziomu do sumy rozmiarów zbiorów uczących węzłów z poprzedniego. Ponadto czas uczenia zależy w przybliżeniu sześciennie od rozmiaru przestrzeni wejściowej oraz rozmiaru przestrzeni wyjściowej. Natomiast, co oczywiste, zależy jedynie liniowo od liczby iteracji oraz, co istotne, również jedynie *liniowo* od liczby przykładów w zbiorze (o ile liczba ta jest na tyle duża, aby liczba neuronów ukrytych sieci neuronowych nie była ograniczana przez liczbę przykładów uczących – zob. podrozdział 4.2).

4.4.3. Oszacowanie złożoności uzyskania odpowiedzi dla podstawowej wersji

Uzyskanie odpowiedzi aproksymacji w węźle wewnętrznym i dla zbioru $|\mathbf{T}_i|$ wymaga:

1. Uzyskania wyniku aproksymacji składowej:
   $O(w|\mathbf{T}_i|) = O\left(\left(\left\lceil\sqrt{p+r^2}\right\rceil(p+r+1)+r\right)|\mathbf{T}_i|\right)$.
2. Przekształcenia przestrzeni przez transformację znalezioną przez PCA:
   $O(|\mathbf{T_i}|(r+p)^2)$.
3. Obliczenia stopni przynależności przykładów do klastrów otrzymanych z algorytmu rozmytych c-średnich – $O(k(r+p)|\mathbf{T_i}|)$.
4. Progowania – $O(k|\mathbf{T_i}|)$.
5. Uzyskania odpowiedzi węzłów potomnych.
6. Obróbki końcowej – $O(a(r+p)|\mathbf{T_i}|)$.

Po uwzględnieniu rzeczywistych zależności między liczbą przykładów, wag itp., jak przy szacowaniu złożoności uczenia, złożoność uzyskiwania wyniku wynosi:

$$O\left(\left((k+r+p)(r+p)+\left(\left\lceil\sqrt{p+r^2}\right\rceil(p+r+1)+r\right)\right)|\mathbf{T_i}|\right)$$

W liściach jest to tylko:

$$O\left(\left(\left\lceil\sqrt{p+r^2}\right\rceil(p+r+1)+r\right)|\mathbf{T_i}|\right)$$

Ogółem w drzewie będzie to

$$O\left(\left(\frac{a^n-1}{a-1}(k+r+p)(r+p)+\frac{a^{n+1}-1}{a-1}\left(\left\lceil\sqrt{p+r^2}\right\rceil(p+r+1)+r\right)\right)|\mathbf{T}|\right)$$

Podobnie jak w przypadku uczenia, uzyskiwanie wyniku zależy wykładniczo (z podstawą a) od wysokości drzewa, a liniowo od wielkości zbioru testowanego. Także

liniowo zależy od poziomu rozgałęzienia drzewa. Tym razem nie są wymagane iteracje, a zależności od liczby wymiarów przestrzeni są w przybliżeniu kwadratowe.

4.4.4. Podsumowanie

Czas uzyskiwania wyników aproksymatora w wersji podstawowej przy niewielkich głębokościach drzewa jest porównywalny z analogicznym czasem dla sieci neuronowej o kilkudziesięciu neuronach ukrytych. Ze względu na zależność wykładniczą czasów uczenia i uzyskiwania wyników od wysokości utworzonego drzewa praktyczne są tylko niewielkie jego wysokości. Silna zależność złożoności obliczeniowej metody od liczby wymiarów wejściowych ogranicza zastosowanie do co najwyżej umiarkowanych liczb cech wejściowych oraz wyjściowych. Sugeruje to każdorazowy rozdział zadania o wyjściu wielowymiarowym na zadania o wyjściach o małej (najlepiej równej 1) liczbie wymiarów. Pamiętać przy tym trzeba, że wiele zadań, np. te wymienione w rozdziale 5, ma jedną zmienną wyjściową. Należy zauważyć, że zależności od wymiarowości przestrzeni zmniejszyłyby się bardzo, gdyby użyć mniej od nich zależnego algorytmu uczenia sieci (mogłoby to jednak wymagać podniesienia liczby iteracji) lub mniej od nich zależnego algorytmu wyznaczania liczby neuronów ukrytych sieci.

5. Wyniki eksperymentalne dla ogólnie znanych zadań

W celu oceny przydatności metody opisanej w niniejszej rozprawie do celów praktycznych została ona przetestowana na 21 zbiorach danych ogólnie dostępnych (niniejszy rozdział) oraz w problemie krótkoterminowego przewidywania profilu zużycia energii elektrycznej (rozdz. 6). Zbiory danych wykorzystane w opisanych poniżej zadaniach pochodzą przede wszystkim z repozytoriów UCI [63] i DELVE [83], i są częścią tzw. "Weka Regression Datasets" [8], większość wskazuje (w chwili pisania niniejszej pracy niedostępną) stronę http://www.ncc.up.pt/ ltorgo/Regression/DataSets.html jako pierwotne źródło. Użycie ogólnie dostępnych zbiorów danych ma tę podstawową zaletę, że otrzymane wyniki można łatwo zweryfikować. Ponadto ich dostępność oraz ich wykorzystanie w literaturze pozwalają porównywać wyniki rozwiązań opisanych w różnych pracach.

Wykorzystane zostało 21 zadań, które można podzielić na 3 grupy: dane sztuczne z jawną funkcją zadaną (zadania 1-3), zadania polegające na powtórzeniu wyników symulacji wykonanej z użyciem dedykowanych systemów, nieposiadające jawnie wyspecyfikowanej odpowiedzi (zadania 4-9) i wreszcie zadania oparte na rzeczywistych danych pomiarowych (10-21). W tej ostatniej grupie można wyróżnić podgrupę zbiorów małych ($\leqslant$ 500 przykładów, zbiory 16-21). Pierwsza grupa jest istotna, ponieważ nie tylko znana jest z całkowitą pewnością funkcja szukana, ale też wiadomo, jak wiele można się o tej funkcji z danego zbioru nauczyć. Zadania oparte na zbiorach rzeczywistych natomiast są szczególnie ważne, ponieważ to właśnie działanie w rzeczywistych problemach jest ostatecznym testem uczenia maszyn i sztucznej inteligencji.

5.1. Zadania

Zgodnie z sugestiami zawartymi w [60] numeryczne dane wejściowe zostały znormalizowane tak, aby wszystkie miały wariancję równą 1 i średnią równą 0. Dane wyjściowe zostały znormalizowane liniowo tak, aby mieściły się dokładnie w zakresie

0.05 - 0.95. Zmienne dyskretne zazwyczaj zostały zamienione na zestawy zmiennych binarnych (z wyjątkiem zadania 21).

W nawiasach po nazwie zadania zostały wymienione skróty nazw zbiorów funkcjonujące w literaturze anglojęzycznej, które powinny pozwolić na ich łatwe wyszukanie, a które ze względu na swoją zwięzłość zostały użyte także w tablicach z wynikami (w kilku przypadkach skrócone jeszcze bardziej).

1. Płaszczyzny dwuwymiarowe (*2dplanes*). Zadanie sztuczne, docelowa funkcja ma postać:

$$f(x) = \begin{cases} 3 + 3x_2 + 2x_3 + x_4 + \mathcal{N}(0,1), & \text{dla } x_1 = 1 \\ -3 + 3x_5 + 2x_6 + x_7 + \mathcal{N}(0,1), & \text{dla } x_1 = -1 \end{cases}$$

   gdzie $\mathcal{N}(0,1)$ oznacza rozkład Gaussa. Przy czym x_1 przyjmuje tylko te dwie wartości z równym prawdopodobieństwem, a pozostałe zmienne wejściowe przyjmują z jednakowym prawdopodobieństwem wartości ze zbioru {-1,0,1}. Zbiór, wymieniony m. in. w [19], zawiera 40768 przykładów i 10 atrybutów wejściowych, przy czym ostatnie 3 są nieużywane.
2. Zadanie sztuczne "Friedmana" (*friedman*). Użyte np. w [38], choć tam wspomniany jest jeszcze wcześniejszy artykuł. Również zadanie sztuczne, prawdziwa funkcja ma postać: $f(x) = 10\sin(\pi x_1 \cdot x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \mathcal{N}(0,1)$. Zbiór danych zawiera także 40768 przykładów oraz 10 atrybutów wejściowych, z czego tylko pierwsze 5 jest wykorzystanych. Zbiór wykorzystywany dość często do testowania różnych rozwiązań, np. drzew decyzyjnych, metod statystycznych, ale także metod wzmacniania gradientowego (gradient boosting) i innych [16, 38].
3. Zadanie sztuczne, o kilku "załamaniach" (*mv*). Funkcja

$$f(x) = \begin{cases} 35 - \dfrac{1}{2}x_4, & \text{dla } x_2 > 1 \\ 10 - 2x_1 & \text{wpp., o ile } -2 \leqslant x_4 \leqslant 2. \\ 3 - x_1/x_4 & \text{wpp., o ile } x_7 = \textbf{yes} \\ x_6 + x_1 & \text{wpp., o ile } x_8 = \textbf{normal} \\ x_1/2 & \text{wpp.} \end{cases}$$

   Zbiór ten również składa się z 10 atrybutów wejściowych i 40768 przykładów. 5 atrybutów jest losowanych z rozkładu jednostajnego na różnych przedziałach. Wśród pozostałych atrybutów 3 (x_3, x_7, x_8) są nominalne, a 2 (x_4 i x_6) mają inne formy i są częściowo zależne od poprzednich.
4. Symulacja kolejek w banku (*bank8fm*). Jeden z rodziny zbiorów powstałych w wy-

niku prostej symulacji zachowań klientów banków. Klienci określeni są kilkoma parametrami, takimi jak miejsce zamieszkania (używane w celu określenia preferencji względem danego banku wynikających z jego dostępności), stopień skomplikowania sprawy, z którą przychodzą, czy cierpliwość. Celem przewidywania jest ułamek osób, które opuszczają bank ze względu na zbyt długie kolejki [83]. Symulacja ta została przeprowadzona bezpośrednio w celu wygenerowania rodziny zbiorów danych, umieszczonych następnie w repozytorium DELVE [83].

5. Symulacja steru wysokości (*elevators*). Dane te powstały w wyniku symulacji zachowania samolotu F16 [26, 27, 33]. Zadane manewry były wykonywane na symulatorze ACM przez autora oryginalnych notatek – Rui Camacho. Na podstawie pomiarów 18 zmiennych wykonanych w czasie tych symulacji, takich jak: prędkość wznoszenia/opadania, wysokość, cel wysokościowy, przechyły samolotu oraz pochodne po czasie niektórych z tych wielkości, starano się odtworzyć decyzje dotyczące ustawienia steru wysokości [26, 27]. Zbiór zawiera 8752 przykładów.
6. Symulacja różnicowa steru wysokości (δ *elevators*). Jak w 5, ale zamiast wartości bezwzględnych odtwarzane były zmiany nastaw steru wysokości. Wykonana została również selekcja atrybutów, wskutek czego liczba zmiennych wejściowych zmniejszyła się do 6. Liczba przykładów wynosi 7129.
7. Symulacja różnicowa ustawienia lotek (δ *ailerons*). Podobnie jak w 6, ale celem były zmiany w nastawie lotek. Liczba zmiennych wejściowych wynosi 5. Liczba przykładów jest równa 7129.
8. Symulacja prędkości kątowej ramienia robota (*puma8NH*). Dane pochodzą z symulacji zachowania ramienia robota Puma 560. Dany zbiór jest jednym z rodziny ośmiu zbiorów dotyczących tego problemu umieszczonych w repozytorium DELVE [83]. Na podstawie ośmiu danych wejściowych, w rodzaju kątów, prędkości, ew. innych danych członów ramienia robota, należy wyznaczyć prędkość kątową jednego z ramion.
9. Symulacja ustawienia ramienia robota (*kin8nm*). Zbiór danych wygenerowany w 1996 r. przez Zoubin Ghahramani za pomocą ówczesnego pakietu języka Matlab z zakresu robotyki: "The Matlab Robotics Toolbox". Dotyczy obliczania pozycji końcówki ramienia robota na podstawie danych o względnej pozycji ramion z dodanym szumem [41]. Ten zbiór również jest jednym z rodziny zbiorów danych dostępnych w DELVE [83]. Zawiera 8192 przykładów. Końcówka *8nm* oznacza 8 zmiennych wejściowych, dużą nieliniowość, ale średni poziom szumów.
10. Wyznaczanie wieku słuchotek (*abalone*). Zbiór danych dotyczący gatunku jadalnych mięczaków z rodziny słuchotek, *Haliotis rubra*. Celem jest określenie wieku mięczaka, badanego bardziej bezpośrednio za pomocą pracochłonnego liczenia

pierścieni wzrostu z użyciem mikroskopu, na podstawie innych pomiarów, np. rozmiarów, masy całego okazu oraz różnych jego części (muszli, mięsa) [68, 70]. Zawiera 4177 przykładów, 7 zmiennych wejściowych.

11. Wycena (grup) domów w Kalifornii (*california housing*). Zbiór dotyczący cen domów umieszczony w repozytorium StatLib [1], wymieniany m.in. w [54]. Celem jest wyznaczenie mediany ceny domu na pewnym obszarze (*census block group* wg terminologii United States Census Bureau – w przykładzie średnia liczba mieszkańców w takiej "grupie kwartałów cenzusowych" wyniosła 1425.5). Dane pochodzą ze spisu powszechnego, każdy przykład jest zbiorczą charakterystyką jednego obszaru. Osiem zmiennych wejściowych obejmuje m.in. liczbę ludności, liczbę pomieszczeń i sypialni w obszarze, liczbę domów, medianę wieku budynków. Zbiór zawiera 20640 przykładów - wszystkie obszary wspomnianego typu z Kalifornii z roku 1990.
12. Wycena (grup) domów, wersja łatwiejsza (*house8L*). Dane pochodzące z repozytorium DELVE [83]. Tak jak i dane zbioru powyżej pochodzą ze spisu powszechnego, ale używana jest inna wielkość jednostkowego obszaru (*State-Place*) i inny obszar. Ten zbiór obejmuje 8 zmiennych wejściowych, niektóre podobne do użytych w poprzednim zbiorze, ale są też dane demograficzne dotyczące wieku i rasy mieszkańców i właścicieli. W zbiorze jest 22784 przykładów. Celem również jest przewidywanie mediany cen domów na danym obszarze. Litera L w oznaczeniu odpowiada mniejszej trudności problemu ze względu na lepszą selekcję zmiennych [83].
13. Wycena (grup) domów, wersja trudniejsza (*house16H*). Jak powyższe, ale 16 zmiennych wejściowych i 22784 przykładów. Litera H oznacza wysoką trudność ze względu na niezbyt dobrą selekcję zmiennych [83].
14. Przewidywanie czasu spędzonego przez procesor w trybie użytkownika (*cpu act*). Zbiór z repozytorium DELVE [83]. Dane dotyczą aktywności uniwersyteckiego systemu komputerowego z rodziny Sparc. Na podstawie 21 ciągłych atrybutów wejściowych w rodzaju: częstość odczytów i zapisów pomiędzy pamięcią systemu, a pamięcią użytkownika, liczba wywołań systemowych na sekundę (osobno też odczytów i zapisów), liczba wywołań nowych procesów, częstości różnych operacji dotyczących stron pamięci, długość kolejki procesów, wielkość wolnej pamięci, należy obliczyć, jaki procent czasu procesor spędzi w trybie użytkownika. Zawiera 8192 przykładów.
15. Przewidywanie czasu spędzonego przez procesor w trybie użytkownika (*cpu_small*). Jak zbiór 14, ale w przewidywaniu nie są dostępne dane dotyczące stronicowania.
16. Szacowanie zużycia paliwa samochodów (*autoMpg*). Zbiór danych dotyczący zu-

życia paliwa (w milach na galon) samochodów osobowych używanych w USA [63, 81]. Ma 5 zmiennych wejściowych: liczba cylindrów, moc, pojemność silnika, masa, przyśpieszenie. Zmienne dotyczące pochodzenia i marki, obecne w oryginalnym zbiorze, zostały usunięte. Zawiera 398 przykładów.

17. Wycena samochodów (*auto*). W tym przypadku na podstawie różnych danych dotyczących samochodu (wymiary, spalanie itp., ale także utrata wartości wg ubezpieczycieli) należy przewidzieć cenę samochodu. Zbiór zawiera zaledwie 159 przykładów, ale 14 zmiennych wejściowych. Pochodzi z repozytorium UCI [63], użyty np. w [81].
18. Przewidywanie poziomu markera insuliny (*diabetes*). Zbiór jest bardzo mały: 43 przykłady, 2 zmienne wejściowe. Na podstawie wieku pacjenta i niedoboru zasad w serum jego krwi należy przewidzieć poziom markera insuliny – peptydu C [47].
19. Wycena (grup) domów w Bostonie (*boston housing*). Na podstawie 12 zmiennych wejściowych typu: liczby przestępstw w jednostce czasu, zanieczyszczenia powietrza, wielkości budynków, liczby nauczycieli na ucznia w miejscowości należy przewidzieć medianę (z obszaru nazywanego *Census-tract*) ceny domów na przedmieściach Bostonu [45]. Zawiera 506 przykładów [63, 81, 83].
20. Przewidywanie wydajności procesora (*m_cpu*) [63]. W tym zadaniu na podstawie 6 zmiennych wejściowych, dotyczących czasu trwania cyklu procesora komputerowego (z końca lat 80), wielkości pamięci głównej i cache oraz opublikowanej wydajności, należy przewidzieć wydajność rzeczywistą, testowaną w oryginalnym artykule.
21. Przewidywanie czasu działania serwomechanizmu (*servo*). Dotyczy przewidywania czasu działania serwomechanizmu ze względu na dwa stopnie wzmocnienia oraz wybór rodzaju śruby (5 możliwości) i silnika (5 możliwości) – w sumie 4 zmienne wejściowe. Zawiera 167 przykładów [80, 81]. W tym przypadku dyskretne zmienne oznaczające wybór śruby i silnika zostały bez zmian – nie zostały zbinaryzowane, ponieważ znacząco podniosłoby to wymiarowość tego niewielkiego zbioru.

5.2. Użyte algorytmy

W tej części eksperymentów zostały użyte następujące wersje proponowanej metody:

1. Z ustawieniem maksymalnej głębokości drzewa na 3 (co oznaczało 4 poziomy) oraz błędu granicznego na 0.

2. Z ustawieniem maksymalnej głębokości drzewa na 3 oraz błędu granicznego na połowę błędu aproksymacji takiej, jak umieszczona w korzeniu (uzyskanego jako średnia z trzech prób).
3. Z ustawieniem maksymalnej głębokości drzewa na 3 oraz błędu granicznego na 0, ale nie używająca wag przy grupowaniu w metodzie opisanej w podrozdziale 4.1.

Dla porównania zostały przetestowane również poniższe algorytmy:

1. "Dobra" sieć neuronowa. Sieci były identyczne jak te w węzłach opisywanego rozwiązania, za wyjątkiem liczby neuronów ukrytych, która była dobierana. Minimalną testowaną liczbą była liczba zgodna z oszacowaniem podanym w punkcie 4.2 – taka jak używana dla aproksymacji w korzeniu tworzonego drzewa. Maksymalna liczba była mniejszą z: 7-krotności liczby minimalnej (żeby nie uczyła się dużo dłużej niż całe 4-poziomowe drzewo) oraz liczby takiej, żeby liczba przykładów na jedną wagę była nie mniejsza niż 2. Aż do 4-krotności liczby minimalnej były testowane wszystkie kolejne liczby neuronów, powyżej, do 6-krotności, była testowana co druga (dla zmniejszenia wymaganego czasu), a powyżej co trzecia. Optymalna w tych warunkach liczba neuronów ukrytych była wybierana przez 3-krotne powtórzenie 10-krotnej walidacji na całym zbiorze dla każdej liczby. Zamieszczone wyniki pochodzą natomiast z kolejnych 10 cykli 10-krotnej walidacji wykonanych dla najlepszej liczby.
2. Komitet. Sieciami bazowymi były sieci neuronowe identyczne jak w korzeniu proponowanego rozwiązania, o liczbie neuronów ukrytych zgodnej ze wzorem (4.1) z podrozdziału 4.2. Liczba sieci została tak dobrana (40), aby suma liczności zbiorów uczących tych sieci była w dużym przybliżeniu równa sumie liczności zbiorów uczących rozwiązań składowych w węzłach. Jako odpowiedź komitetu została wybrana średnia – wstępne eksperymenty ze średnią ważoną nie wykazały poprawy.
3. Agregacja modeli ze zmianą zbiorów uczących *bagging* [16] opisana pokrótce we wstępie (1.4.3, s. 7). Elementami składowymi były także sieci neuronowe takie same jak w powyższych punktach (zwykle używane są tu drzewa regresji).
4. Wzmacniane gradientowe (*gradient boosting*, nazywane także "Additive Regression" [37] wspomniane w podrozdziale 1.4.2, s. 5). Elementy składowe były takie same jak w powyższych punktach. Zostały wypróbowane następujące wartości parametru skalowania: 0.05, 0.1, 0.2, 0.3, 0.5. Procedura testowa była podobna jak w punkcie 1: najpierw powtórzono 3-krotnie 10-krotną walidację na całym zbiorze, aby wybrać parametr, a później wybrany parametr testowano dokładniej 10 razy 10-krotną walidacją. Tak, jak w punkcie 2, zostało użytych 40 sieci.

Trzy z modeli wybranych do porównań, podobnie jak proponowane rozwiązanie, łączą wiele prostych rozwiązań w jedno, potencjalnie dokładniejsze. Właśnie to podobieństwo oraz dostępność implementacji była kluczem przy ich wyborze. Używały one takich samych elementów składowych jak proponowane rozwiązanie, a ich rozmiar i czasochłonność miały być zbliżone do proponowanego rozwiązania, dzięki czemu wyniki eksperymentów mogą stanowić podstawę do oszacowania potencjału rodzajów połączeń prostych sieci w jedno większe i dokładniejsze rozwiązanie.

Należy zauważyć, że parametry kontrolne proponowanej metody oraz *bagging*u i komitetu *nie były dopasowywane do poszczególnych zadań* w żaden sposób. Natomiast w przypadku sieci neuronowej oraz wzmacniania gradientowego testowano różne wartości jednego, potencjalnie najważniejszego parametru i ostatecznie raportowano rozwiązanie, które okazało się najlepsze we wstępnej walidacji. Walidacja ta używała całego zbioru danych, czyli wybór tych rozwiązań zależał pośrednio od danych testowych. Takie postępowanie mogło wprowadzić nierównowagę w testach na korzyść tych dwu rozwiązań, tj. sieci neuronowej i wzmacniania gradientowego. Ze względu jednak na wagę owych parametrów i trudność w ich oszacowaniu a-priori uznano opisaną metodę za dobry kompromis, o ile porównujemy głównie rozwiązanie proponowane. Naturalnie, w tych dwu rozwiązaniach można dostosowywać nawet więcej parametrów kontrolnych, np. dobierać specjalnie sieci we wzmacnianiu gradientowym albo zmieniać algorytmy lub współczynniki uczenia czy liczbę warstw w sieci neuronowej. To samo można jednak powiedzieć o rozwiązaniu proponowanym, w którym nie dostosowywano ani jednego, nawet podstawowego, parametru kontrolnego. Dla komitetów i agregacji modeli ze zmianą zbiorów uczących (*bagging*) testowano tylko maksymalną dopuszczalną liczbę (i jeden rodzaj) sieci, ponieważ jej zmniejszenie nie powinno powodować wzrostu dokładności rozwiązań.

5.3. Implementacja proponowanego rozwiązania

Algorytmy zostały zaimplementowane w języku Java. Wykorzystana została w nich biblioteka WEKA (*Waikato Environment for Knowledge Analysis*) [43]. Implementacje dwóch algorytmów uprzednio istniejących: "Bootstrap Aggregating" i wzmacniania gradientowego pochodzą z tej biblioteki (ew. z drobnymi modyfikacjami). Implementacja komitetu, tak jak i proponowanej metody została wykonana przez autora niniejszej pracy. Dla celów m.in. tej pracy powstały również implementacje algorytmów pomocniczych: wykorzystywanych we wszystkich rozwiązaniach sieci neuronowych uczonych algorytmem Levenberga-Marquardta [97] (zob. punkt

4.3.3) oraz wykorzystywanego w proponowanym rozwiązaniu rozmytego c-średnich [72] (opisanego pokrótce w punkcie 4.3.1).

Implementacja pozwala na zrównoleglenie obliczeń. Dostępna jest pula wątków Javy o zadanej przez użytkownika wielkości. Pojedynczym zadaniem jest uczenie jednego węzła (uczenie rozwiązania składowego i tworzenie zbiorów uczących dla węzłów potomnych).

Implementacja metody Levenberga-Marquardta, zgodnie z [97], nie przechowuje całego jakobianu w pamięci (jak niektóre z dostępnych implementacji), w zamian za to używa ona pamięci proporcjonalnej do iloczynu sumy liczby węzłów ukrytych i wyjściowych oraz liczby przykładów dla przechowywania pośrednich wyników obliczeń dla ich późniejszego wykorzystania. Dzięki temu, że użyte sieci są nieduże, wymagania pamięciowe są umiarkowane, mniejsze niż wymagania implementacji "pamięciowo-naiwnej" [97].

5.4. Procedura testowa i miary błędu

W testach została użyta metoda 10-krotnej walidacji krzyżowej ze stratyfikacją: zbiór był dzielony na 10 części tak, aby średnia odpowiedzi w każdej części była zbliżona. Następnie na 9 częściach algorytm był uczony a na jednej testowany, co powtarzano 10 razy, tak aby dane rozwiązanie zostało przetestowane na każdej części. Całą procedurę powtórzono 10 razy, – w sumie dla każdego zadania wykonano 100 cykli nauki i testowania.

W tablicach uwzględniono przede wszystkim znormalizowany pierwiastek ze średniego błędu kwadratowego (*NRMSE, normalized root-mean squared error*).

$$NRMSE = \sqrt{\frac{\sum_i^{|S|} ||\hat{f}(x_i) - y_i||^2}{\sum_i^{|S|} ||y_i - \overline{Y}||^2}} \tag{5.1}$$

gdzie S to dany zbiór, x_i to i-ty wektor wejściowy, y_i wyjściowy, $\hat{f}$ to odpowiedź rozwiązania, $\overline{Y}$ to średnia po wszystkich wektorach wyjściowych, $||\cdot||$ to norma Euklidesowa.

Miara ta zwiększa wagę dużych błędów i z definicji jest zgodna z błędem średniokwadratowym. Jest wyrażona w jednostkach, w których jest wyrażony problem (w odróżnieniu od błędu kwadratowego albo logarytmicznego), a więc dość intuicyjna. Zapisano średni błąd ze wszystkich 10 powtórzeń 10-krotnej walidacji ($\overline{NRMSE}$) oraz odchylenie standardowe w tych dziesięciu próbach ($\sigma NRMSE$).

Dla porównania ogólnego potencjału rozwiązań potrzebna była miara podsu-

mowująca efektywność na bardzo odmiennych zbiorach danych. Taką rolę mógłby pełnić znormalizowany pierwiastek ze średniego błędu kwadratowego, ale ponieważ odchylenie standardowe wartości wyjść od średniej, uwzględniane przez tę miarę, nie jest jedyną miarą skomplikowania zadania, zdecydowano się w tej pracy przyjąć miarę jeszcze bardziej odporną na różnice pomiędzy zadaniami – rangi. W tym przypadku, użyta jest specjalnie utworzona miara nazywana tu "rangą istotną".

Przez "istotną rangę" dla rozwiązania i rozumiem powiększoną o 1 liczbę rozwiązań j, takich że dla danego zadania $(\overline{\text{NRMSE}_j} + \sigma\text{NRMSE}_j) * 1.03 < \overline{\text{NRMSE}_i} - \sigma\text{NRMSE}_i$. A zatem: oprócz tego, że średnie z dziesięciu prób powinny być odległe od siebie przynajmniej o dwa odchylenia standardowe (po jednym z każdego rozwiązania), to musi zostać zachowany dodatkowy "trzyprocentowy" odstęp. Dzięki temu w zasadzie można powiedzieć, że jeśli jakieś rozwiązanie ma niższą rangę, to znaczy, że istnieje dość duże prawdopodobieństwo, że jego średni błąd jest lepszy przynajmniej o 3%. Konkretne wartości są tu w zasadzie arbitralne, można sobie wyobrazić całą rodzinę miar "δ, ϵ istotnych", w których $(\overline{\text{NRMSE}_j} + \delta\sigma\text{NRMSE}_j) * (1 + \epsilon) < \overline{\text{NRMSE}_i} - \delta\sigma\text{NRMSE}_i$, ale użyte wartości ($\delta = 1, \epsilon = 0.03$) można uznać za suboptymalne. Wspomniane 3% jest minimalną różnicą błędów, którą można uznać za mającą znaczenie praktyczne (choć to, oczywiście, zawsze zależy od konkretnego zadania). Jest to też, w zaokrągleniu do pełnych procent, uśredniona po wszystkich zadaniach różnica pomiędzy NRMSE najlepszego rozwiązania dla danego zadania i NRMSE drugiego w kolejności.

Używane są także, przede wszystkim w dodatkach:

- Średni błąd na moduł, (zob. wzór (2.4), s. 14). Podano średnią z 10 prób i odchylenie standardowe.
- Pierwiastek ze średniego błędu kwadratowego (nienormalizowany):

$$RMSE = \sqrt{\frac{\sum_i^{|X|} ||g(x_i) - y_i||^2}{|X|}} \tag{5.2}$$

  Oprócz średniej i odchylenia standardowego z 10 prób wpisano także $\pm$, czyli maksymalne odchylenie pojedynczej wartości od średniej (z 10 prób).
- Przeskalowany pierwiastek ze średniego błędu kwadratowego:

$$SRMSE = \sqrt{\frac{\sum_i^{|X|} ||(g(x_i) - y_i) \cdot 2s||^2}{|X|}} \tag{5.3}$$

  gdzie s to współczynnik skali – wektor odwrotności różnic między maksymalnymi

i minimalnymi współrzędnymi (na każdej pozycji) w wektorach wyjściowych. Odpowiada przeskalowaniu zmiennej wyjściowej tak, aby mieściła się pomiędzy -1 i 1, co jest czasem używane. Oczywiście można z tej wartości łatwo otrzymać wersję dla skalowania do przedziału $[0, 1]$.

5.5. Wyniki eksperymentów i ich omówienie

5.5.1. Przykładowy przebieg uczenia

W niniejszym podrozdziale przedstawiony zostanie przykładowy przebieg algorytmu uczenia, czyli tworzenia drzewa hierarchicznej aproksymacji, opisanego w ogólnej formie w punkcie 2.4.2 (s. 22), z poszczególnymi komponentami opisanymi w poprzednim rozdziale. W przebiegu eksperymentów z niniejszego rozdziału zostało utworzonych 6300 drzew. Dla celów poglądowych opisane zostanie jedno z drzew utworzonych dla zadania nr 19 (*boston housing*) i sposobu tworzenia aproksymacji nr 2, czyli docelowym błędzie ustalonym na połowę błędu korzenia.

Na początku na podstawie średniej z 3 prób uczenia aproksymacji o takiej strukturze, jaka docelowo była użyta w korzeniu, docelowy średni błąd na moduł został automatycznie ustalony na 0.9. Liczba neuronów ukrytych w korzeniu została automatycznie ustalona na 4. Sieć neuronowa w ostatecznie użytym korzeniu, z błędem na moduł zaledwie 1.55, osiągnęła wynik lepszy niż te, które posłużyły do ustalenia progu. Transformacja przed klastrowaniem (zob. pkt 1 algorytmu z podrozdziału 4.1, s. 43) wybrała 7 kierunków (tab. 5.1). Być może ze względu na specyfikę skalowania, wyjście (pierwszy wiersz) miało niższe wagi, niż większość pozostałych zmiennych.

Klastrowanie w korzeniu w połączeniu z progowaniem (opisanym w punkcie 4.3.2) podzieliło zbiór uczący korzenia na 5 podzbiorów o licznościach 368, 181, 229, 361, 223, przy liczności całego zbioru uczącego 455. Tablica 5.2 ilustruje nakładanie się klastrów. Jak widać zbiory c_1 i c_4 były bardzo podobne.

Uczenie węzłów potomnych dla klastrów pierwszego i trzeciego nie powiodło się – dały one na swoich zbiorach uczących (będących jednocześnie zbiorami kompetencji) błąd wyższy niż korzeń. Czwarty węzeł potomny był uczony dwukrotnie (zob. podrozdział 4.2, s. 45). Wszystkie trzy powstałe w ten sposób węzły miały błędy większe niż ustalony próg zatrzymania.

Dla potomków pierwszego z nich algorytm przewidział zbiory uczące o następujących rozmiarach: 132, 137, 79, 132, 57. Aproksymacja na pierwszym z nich nie została ostatecznie uwzględniona ze względu na zbyt duży błąd, a ostatni węzeł miał błąd mniejszy niż założony próg zatrzymania. W pozostałych węzłach nauka

Tablica 5.1. Kierunki wybrane przez PCA przed klastrowaniem wyników w korzeniu przykładowego drzewa.

v_1	v_2	v_3	v_4	v_5	v_6	v_7
0.046	-0.047	0.067	0.003	0.051	-0.010	0.002
-0.251	0.292	0.262	-0.048	-0.070	-0.182	-0.776
0.259	0.304	0.307	-0.027	-0.307	-0.380	0.238
-0.345	-0.114	-0.012	-0.014	0.014	-0.093	0.361
0.002	-0.453	0.272	-0.774	-0.308	0.145	-0.047
-0.341	-0.229	0.108	0.166	-0.072	-0.168	0.198
0.197	-0.163	0.560	0.135	0.501	0.074	-0.076
-0.315	-0.321	-0.033	0.160	0.040	-0.048	-0.137
0.317	0.351	-0.049	-0.164	-0.174	-0.004	0.108
-0.317	0.262	0.279	-0.194	0.180	-0.162	0.120
-0.333	0.232	0.218	-0.135	0.123	-0.223	0.298
-0.202	0.334	-0.342	-0.424	0.441	0.299	0.012
0.210	-0.241	-0.356	-0.246	0.322	-0.763	-0.122
-0.313	0.068	-0.246	0.055	-0.411	-0.097	-0.125

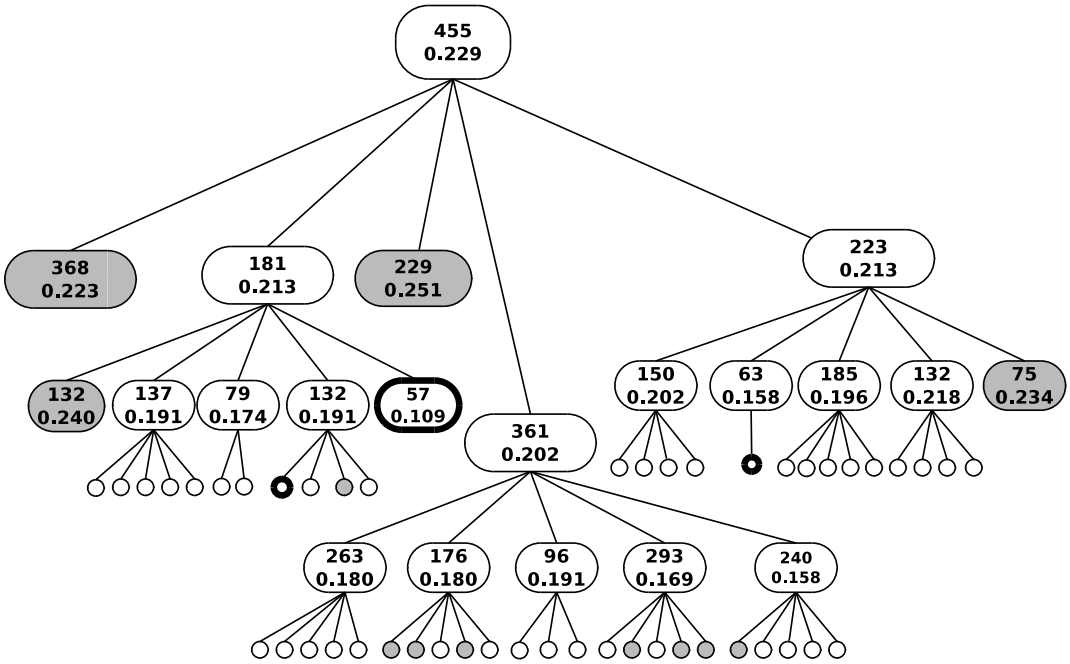
Tablica 5.2. Nakładanie się zbiorów uczących dla bezpośrednich potomków korzenia w przykładowym drzewie.

	c_1	c_2	c_3	c_4	c_5
c_1	368	122	2	357	195
c_2	122	181	149	126	71
c_3	2	149	229	6	115
c_4	357	126	6	361	184
c_5	195	71	115	184	223

była kontynuowana. Dla potomków drugiego rozmiary zbiorów wyniosły 263, 176, 96, 293, 240, wszystkie miały błędy na swoich zbiorach kompetencji niższe niż ich rodzic, ale wyższe niż próg zatrzymania, a więc algorytm był w nich kontynuowany. Liczności zbiorów uczących potencjalnych potomków trzeciego bezpośredniego potomka korzenia wyniosły z kolei 150, 63, 185, 132, 75. Dla ostatniego z tych zbiorów węzeł nie został jednak utworzony z powodu zbyt wysokiego błędu. W pozostałych węzłach algorytm był kontynuowany.

Na ostatnim poziomie zostało utworzonych i uwzględnionych 40 aproksymatorów, 8 kolejnych zostało odrzuconych ze względu na zbyt duży błąd, a 12 nie zostało w ogóle utworzonych z powodu zbyt małej liczby przykładów (zob. podrozdział 4.2).

Ostatecznie powstała struktura przedstawiona jest na rysunku 5.1. Oprócz ogólnej struktury uwzględniono na nim rozmiary zbiorów uczących (liczba położona wyżej w obrębie oznaczenia węzła) i błędy NRMSE aproksymacji w danym węźle na jego zbiorze uczącym (liczba położona niżej) dla węzłów z poziomów od pierwszego (korzeń) do trzeciego. Dla węzłów z poziomu czwartego dane te opuszczono z uwagi na czytelność diagramu. Szarym tłem wyróżniono węzły usunięte ze względu na błąd



Rysunek 5.1. Struktura przykładowego drzewa dla zadania *housing*. Liczby w węzłach: rozmiar zbioru uczącego (wyższa), NRMSE (niższa). Szare tło oznacza usunięty węzeł, gruba obwódka – węzeł o błędzie niższym niż docelowy.

aproksymacji na własnym zbiorze uczącym większy niż błąd rodzica na tym samym zbiorze. Pogrubiona krawędź otacza węzły o średnim błędzie na moduł niższym niż docelowy błąd. Należy zauważyć, że błąd NRMSE dla jednego z węzłów jest wyższy od wpisanego błędu NRMSE jego rodzica, a mimo to nie został on odrzucony. Dzieje się tak dlatego, że porównywane są błędy na zbiorze kompetencji dziecka. W przypadku użytej funkcji kompetencji zbiór kompetencji dziecka jest wybrany ze zbioru uczącego rodzica i jest równy zbiorowi uczącemu dziecka.

Wszystkie zbiory uczące aproksymatorów zawierały łącznie 15490 przykładów, z czego 7432 w aproksymatorach odrzuconych. W pierwszej z tych liczb zbiory uczące niektórych zaakceptowanych węzłów, a w drugiej zbiory uczące wszystkich odrzuconych węzłów, są uwzględnione podwójnie, na skutek powtarzania uczenia (zob. punkt 4.2 oraz pkt 4 głównego algorytmu, s. 22). Spośród węzłów istotnie utworzonych 32 węzły były uczone dwukrotnie (gł. na niższych poziomach), a 25 (łącznie z korzeniem) jednokrotnie.

Łącznie wystąpiły 4 próby uczenia sieci o 5 neuronach ukrytych, 14 prób o 4 neuronach ukrytych, 32 o 3, 50 o 2 i 15 o jednym neuronie ukrytym. Te ostatnie odpowiadają regresjom liniowym z jedną funkcją nieliniową nałożoną na końcu. Ponowne uczenie sieci było tu traktowane jako oddzielna próba.

5.5.2. Porównanie wyników testowanych rozwiązań

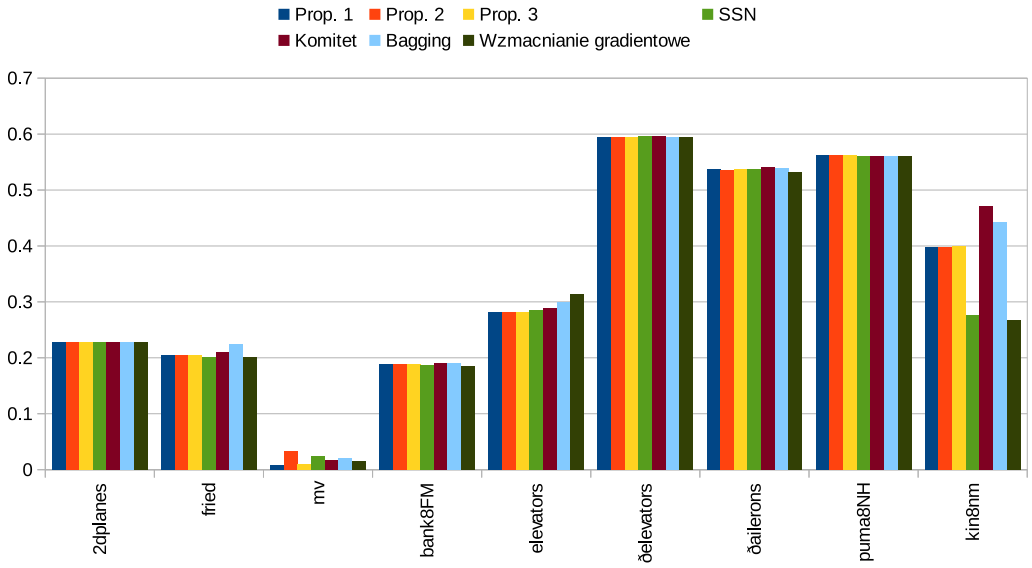
W tablicach 5.3 do 5.7 oraz w dodatku B zostały zaprezentowane wyniki opisanych powyżej eksperymentów. W tablicach 5.3 do 5.7 znajdują się dane dotyczące miary $NRMSE$ oraz rankingi dotyczące wyników na zbiorach testowych. W dodatku B zostały podane inne miary błędów dla zbiorów testowych oraz podstawowe wyniki na zbiorach uczących. Raportowane wyniki i rangi powstały w oparciu o dane o pełnej precyzji (nie użytej w tablicach ze względu na czytelność).

Tablica 5.3. Wyniki na częściach testujących dla proponowanych rozwiązań.

Rozwiązanie:	1 (docelowy błąd 0)		2 ($\frac{1}{2}$ błędu korzenia)		3(bez wag)	
Zadanie	$\overline{\text{NRMSE}}$	σNRMSE	$\overline{\text{NRMSE}}$	σNRMSE	$\overline{\text{NRMSE}}$	σNRMSE
2dplanes	0.227	0.0000	0.227	0.0000	0.227	0.0000
friedman	0.205	0.0002	0.205	0.0002	0.205	0.0002
mv	0.008	0.0003	0.033	0.0169	0.010	0.0002
bank8fm	0.188	0.0001	0.188	0.0003	0.188	0.0003
elevators	0.282	0.0007	0.282	0.0010	0.281	0.0007
δelevators	0.594	0.0005	0.594	0.0003	0.595	0.0003
δailerons	0.537	0.0017	0.536	0.0017	0.537	0.0016
puma8NH	0.561	0.0003	0.562	0.0003	0.562	0.0003
kin8nm	0.397	0.0026	0.398	0.0020	0.399	0.0021
abalone	0.645	0.0020	0.645	0.0017	0.644	0.0015
california	0.480	0.0007	0.481	0.0005	0.481	0.0005
house8L	0.569	0.0013	0.570	0.0022	0.581	0.0023
house16H	0.621	0.0020	0.620	0.0050	0.627	0.0032
cpu_act	0.125	0.0005	0.125	0.0004	0.125	0.0006
cpu_small	0.157	0.0004	0.156	0.0006	0.157	0.0004
autoMpg	0.326	0.0028	0.323	0.0030	0.324	0.0032
auto	0.459	0.0296	0.458	0.0434	0.469	0.0484
diabetes	0.929	0.0319	0.914	0.0377	0.926	0.0318
boston	0.297	0.0078	0.299	0.0096	0.295	0.0064
m cpu	0.357	0.0235	0.367	0.0369	0.376	0.0424
servo	0.247	0.0164	0.244	0.0128	0.244	0.0118
Średnia	0.391		0.392		0.393	

Jak można zauważyć w tablicach 5.3 i 5.4 oraz na rysunkach 5.2 i 5.3 w wielu zadaniach (np. dwuwymiarowe płaszczyzny, zbiory pochodzące z symulacji z wyjątkiem kinematyki ramienia robota *kin8nm*, szacowanie zużycia paliwa *autoMpg*) wiele z porównywanych metod miało bardzo zbliżone wyniki. Może być to skutkiem dużego podobieństwa elementów składowych rozwiązań (zob. podrozdział 5.2).

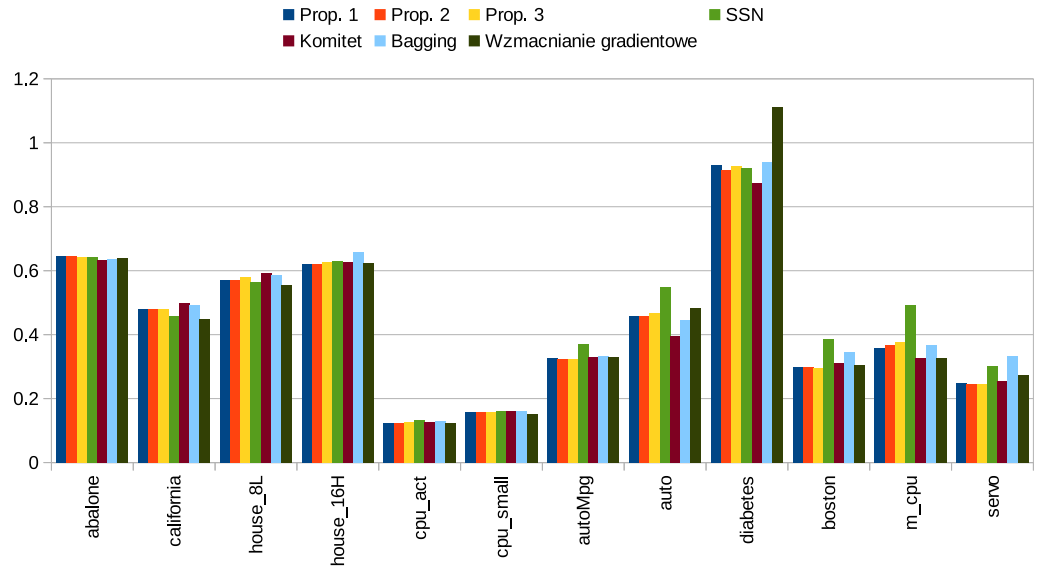
Zgodnie z oczekiwaniami, na większych zbiorach różnice pomiędzy wynikami na zbiorze uczącym i zbiorze testującym były niewielkie, zwykle nie przekraczające



Rysunek 5.2. Porównanie błędu NRMSE na zadaniach sztucznych i powstałych z symulacji.

kilku procent, z istotnym wyjątkiem zbioru *mv*. Zupełnie nie dziwi to w przypadku komitetu. Jest jednak interesujące dla wzmacniania gradientowego, a także w przypadku pierwszego z proponowanych rozwiązań, gdyż jeden z głównych parametrów kontrolujących generalizację ("docelowy błąd" pkt 3 w punkcie 2.4.2) został w istocie wyłączony – ustawiony na 0. Zjawisko to jest w części wynikiem ograniczenia głębokości drzewa, oraz wprowadzenia w uczeniu sieci neuronowej kary za duże wagi (wspomniane w punkcie 4.3.3, s. 48). Częściowo wynika jednak również z cech architektury systemu. Jedną z tych cech może być przypisywanie istotnie niezerowej kompetencji węzłom wewnętrznym w drzewie (zob. 4.1), operującym na większych zbiorach danych, a przez to bardziej ogólnym. Jest to możliwe, ponieważ one też uczą się rozwiązywać podproblem danego problemu, a nie tylko rozdzielać przykłady, jak np. w HME [53]. Inną cechą może być używanie średniej ważonej z ograniczonymi do przedziału [0,1] współczynnikami (zamiast np. arbitralnych, optymalizowanych wag). Pomagać może również to, że węzły uczą się podproblemów głównego problemu – ograniczonych co prawda, ale nie modyfikowanych poprzez np. zmianę funkcji celu jak we wzmacnianiu gradientowym.

Na zbiorach mniejszych różnica pomiędzy zbiorami testowymi i uczącymi jest wyraźnie większa – najlepiej wypadają w tej kwestii *bagging* i komitet. Pomimo iż jeden z parametrów kontrolujących generalizację jest wyłączony i używane są dokładnie te same parametry kontrolne, co na większych zbiorach, proponowane rozwiązania



Rysunek 5.3. Porównanie błędu NRMSE na zadaniach ze świata rzeczywistego.

zachowują różnicę pomiędzy skutecznością na zbiorach uczących i testujących w rozsądnych granicach, choć istotnie szerszych niż w przypadku wspomnianych wyżej rozwiązań (zob. tab. 5.5). W tym przypadku znaczenie miała zasada mówiąca, że każda sieć w drzewie powinna mieć odpowiednią liczbę przykładów do nauki, w razie braku której liczba węzłów ukrytych (czyli również wag) może zostać ograniczona, a w skrajnych przypadkach sieć może nie zostać utworzona (zob. podrozdział 4.2).

Najniższy średni znormalizowany pierwiastek ze średniego błędu kwadratowego osiągnęły rozwiązania proponowane, ale różnice nie były duże: 3.8% pomiędzy najlepszym i najgorszym.

Jak można zauważyć, szczególnie w tablicach z dodatku B (gdyż tam znajduje się nieznormalizowany RMSE), zadania sztuczne o płaszczyznach dwuwymiarowych i *friedman* zostały w kilku przypadkach rozwiązane bardzo dobrze, z błędem mniej niż o 1% większym od teoretycznego optimum - RMSE=1, wynikającego z obecności gaussowskiego szumu o σ 1.

W obliczu tak wielu różnych zadań średni NRMSE ze wszystkich zadań może nie wystarczać do porównania efektów uczenia przy użyciu różnych metod. Dlatego w tablicach 5.6 i 5.7 podano "istotne rangi" (zob. podrozdział 5.4) uzyskane na poszczególnych zbiorach i ich średnią. Ze względu na duże podobieństwo błędów trzech wersji proponowanego rozwiązania, te rozwiązania, które miały większy błąd od jednego z nich, miały zwykle większy błąd niż wszystkie i odpowiednio niższą rangę w tablicy 5.6. Dla oceny proponowanego rozwiązania lepsza jest w takim

Tablica 5.4. Wyniki na częściach testujących dla rozwiązań porównywanych.

Rozw.:	1 SSN		2 (komitet)		3 (bagging)		4 (Wzm. Grad.)	
Zadanie	$\overline{\text{NRMSE}}$	σNRMSE	$\overline{\text{NRMSE}}$	σNRMSE	$\overline{\text{NRMSE}}$	σNRMSE	$\overline{\text{NRMSE}}$	σNRMSE
2dplanes	0.227	0.0000	0.227	0.0000	0.227	0.0000	0.228	0.0000
friedman	0.202	0.0001	0.209	0.0002	0.224	0.0062	0.201	0.0001
mv	0.024	0.0011	0.018	0.0001	0.021	0.0005	0.015	0.0000
bank8fm	0.187	0.0008	0.190	0.0001	0.190	0.0002	0.186	0.0003
elevators	0.285	0.0041	0.289	0.0008	0.299	0.0023	0.313	0.0317
δelevators	0.597	0.0007	0.596	0.0002	0.595	0.0002	0.594	0.0004
δailerons	0.538	0.0029	0.542	0.0000	0.539	0.0010	0.532	0.0014
puma8NH	0.561	0.0005	0.561	0.0001	0.561	0.0001	0.561	0.0004
kin8nm	0.275	0.0015	0.470	0.0005	0.443	0.0019	0.267	0.0013
abalone	0.642	0.0027	0.635	0.0003	0.636	0.0010	0.640	0.0014
california	0.459	0.0009	0.498	0.0003	0.494	0.0008	0.448	0.0014
house8L	0.566	0.0036	0.592	0.0007	0.586	0.0008	0.555	0.0014
house16H	0.631	0.0035	0.626	0.0058	0.657	0.0028	0.624	0.0018
cpu_act	0.133	0.0023	0.126	0.0020	0.131	0.0008	0.124	0.0011
cpu_small	0.161	0.0027	0.160	0.0015	0.162	0.0005	0.150	0.0007
autoMpg	0.370	0.0084	0.330	0.0180	0.333	0.0019	0.329	0.0060
auto	0.550	0.0590	0.394	0.0899	0.446	0.0169	0.482	0.0544
diabetes	0.920	0.0432	0.875	0.1013	0.938	0.0177	1.112	0.0513
boston	0.387	0.0165	0.310	0.0335	0.344	0.0057	0.306	0.0069
m cpu	0.493	0.0483	0.327	0.0636	0.368	0.0112	0.327	0.0331
servo	0.302	0.0251	0.255	0.0304	0.333	0.0071	0.274	0.0060
Średnia	0.405		0.392		0.406		0.394	

Tablica 5.5. Średnia geometryczna NRMSE błędów testowych jako procentów NRMSE na zbiorach uczących.

Rozwiązania:	Proponowane			Istniejące			
Zbiory↓	1	2	3	1(SSN)	2(Kom.)	3(Bag.)	4(WG)
"duże"1-15	103.7	103.7	103.2	105.2	101.4	101.3	107.0
"małe"16-21	204.6	202.9	204.9	188.5	170.9	172.1	258.5
Wszystkie	125.9	125.6	125.6	124.3	117.7	117.9	137.7

razie tablica 5.7, choć jest tam zamieszczone tylko jedno (potencjalnie najlepsze) z proponowanych rozwiązań. Z tablicy 5.6 można natomiast wnioskować, że podstawowa wersja proponowanego rozwiązania miała minimalną przewagę nad pozostałymi dwoma, w szczególności nad metodą nie używającą wag, co sugeruje niewielką ale istniejącą ich przydatność wag klasteryzacji (zob. pkt 2 algorytmu tworzenia dzieci z podrozdziału 4.1, s. 44) dla rozwiązania.

Zgodnie z tablicą (5.7) najniższe średnie "istotne rangi" osiągnęły proponowane rozwiązanie oraz wzmacnianie gradientowe. Porównanie bezpośrednie wskazuje, że

Tablica 5.6. "Istotne rangi" na częściach testujących.

Rozwiązania:	Proponowane			Istniejące			
	1	2	3	1(SSN)	2(Kom.)	3(Bag.)	4(WG)
2dplanes	1	1	1	1	1	1	1
friedman	1	1	1	1	3	7	1
mv	1	4	1	5	4	4	3
bank8fm	1	1	1	1	1	1	1
elevators	1	1	1	1	1	4	1
δelevators	1	1	1	1	1	1	1
δailerons	1	1	1	1	1	1	1
puma8NH	1	1	1	1	1	1	1
kin8nm	3	3	3	1	7	6	1
abalone	1	1	1	1	1	1	1
california	3	3	3	1	6	3	1
house8L	1	1	2	1	5	2	1
house16H	1	1	1	1	1	7	1
cpu_act	1	1	1	5	1	5	1
cpu_small	2	2	2	2	2	2	1
autoMpg	1	1	1	7	1	1	1
auto	1	1	1	2	1	1	1
diabetes	1	1	1	1	1	1	7
boston	1	1	1	7	1	5	1
m cpu	1	1	1	7	1	1	1
servo	1	1	1	4	1	6	3
Średnia	1.24	1.38	1.29	2.48	2.00	2.90	1.48

wzmacnianie gradientowe miało mniejszą "istotną rangę" dla 3, a większa dla 2 zadań, natomiast w 16 zadaniach "istotne rangi" tych dwóch metod były takie same. Dla zadań, dla których "istotne rangi" wzmacniania gradientowego były większe niż "istotne rangi" rozwiązania proponowanego w rozprawie, błędy wzmacniania gradientowego były większe o 84% i 20%. Dla tych zadań, dla których zaszedł przypadek odwrotny, błędy proponowanego rozwiązania były większe o: 49%, 7% i 4%. Jak widać, różnice na korzyść proponowanego rozwiązania są mniej liczne, ale większe, dlatego średni NRMSE (tab. 5.3 i 5.4) proponowanego rozwiązania jest niższy (choć zaledwie o ok 1%) niż wzmacniania gradientowego. Pozostałe rozwiązania mają większe średnie "rangi istotne", chociaż dla kilku zadań różnice pomiędzy wszystkimi testowanymi rozwiązaniami były nieznaczne, czego objawem są równe "rangi istotne".

Uwzględniając czasy z tablicy B.8 z dodatku B, widać, że średni czas uczenia systemu jest wyższy niż czasu uczenia porównywanych rozwiązań – różnice dochodzą do kilkudziesięciu procent. Można zauważyć wyraźną zależność czasu uczenia

Tablica 5.7. "Istotne rangi" przy uwzględnieniu tylko podstawowej wersji proponowanego rozwiązania.

Rozwiązania:	Proponowane	Istniejące			
	1	1(SSN)	2(Kom.)	3(Bag.)	4(WG)
2dplanes	1	1	1	1	1
friedman	1	1	3	5	1
mv	1	4	3	3	2
bank8fm	1	1	1	1	1
elevators	1	1	1	2	1
δelevators	1	1	1	1	1
δailerons	1	1	1	1	1
puma8NH	1	1	1	1	1
kin8nm	3	1	5	4	1
abalone	1	1	1	1	1
california	3	1	4	3	1
house8L	1	1	4	2	1
house16H	1	1	1	5	1
cpu_act	1	3	1	3	1
cpu_small	2	2	2	2	1
autoMpg	1	5	1	1	1
auto	1	2	1	1	1
diabetes	1	1	1	1	5
boston	1	5	1	3	1
m cpu	1	5	1	1	1
servo	1	2	1	4	1
Średnia	1.24	1.95	1.71	2.19	1.24

od liczby wymiarów i liniową od liczby przykładów. Dla niektórych zbiorów system uczył się dużo dłużej niż inne porównywane rozwiązania łączenia wielu zmiennych (nawet trzykrotnie), ale dla kilku innych czasy były zbliżone do pozostałych rozwiązań. Należy jednak pamiętać, że sposób zbierania tych informacji był niezbyt dokładny (zob. nota do tablicy B.8). Czasy uczenia wzmacniania gradientowego nie uwzględniają czasu testowania parametru, który był ok. 1.5 raza większy, od czasu ostatecznego testu. W tablicy w dodatku występuje suma zużycia czasu dla wszystkich procesorów i wszystkich rdzeni. Rzeczywisty czas uczenia proponowanego rozwiązania był krótszy ze względu na zrównoleglenie (zob. podrozdział 5.3). Dla rozwiązania z niezerowym warunkiem stopu, czasy uczenia dla kilku zadań były znacznie niższe, z powodu mniejszej liczby węzłów w drzewie.

Za ogólnie zwiększone zapotrzebowanie na moc procesora odpowiedzialne są w dużej mierze algorytm grupowania oraz powtarzanie uczenia, jeżeli wynik był gorszy (zob. 2.4.2, s. 22). W pierwszym przypadku jednak dotyczy to tylko węzłów

wewnętrznych, które, przy zadanych ustawieniach, odpowiadają za ok. 1/3 sumy zbiorów uczących, a sam algorytm działa szybciej niż uczenie klasyfikatorów w węźle. Powtarzanie uczenia zdarzało się ze zróżnicowaną częstością, ale zwykle dotyczyło liści i, w przybliżeniu, występowało na ok. 1/4 całkowitej sumy liczności zbiorów uczących – mniej przy ustawieniu parametru zatrzymania na wartość niezerową. Z drugiej strony, ze względu na usuwanie poddrzew, drzewa nie muszą być pełne, co obniża liczbę rozwiązań koniecznych do nauczenia. Stąd też biorą się znaczące różnice w stosunkach czasów uczenia proponowanego rozwiązania i rozwiązań porównywanych dla różnych zbiorów.

Biorąc pod uwagę obydwa sposoby porównań, można stwierdzić, że w tym eksperymencie najlepiej sprawdziły się: podstawowa wersja proponowanego rozwiązania oraz wzmacnianie gradientowe, wyprzedzając pozostałe rozwiązania porównywane, w tym sieć neuronową, która ma wyniki nie gorsze niż pojedynczy element składowy rozwiązania (ze względu na procedurę wybierania liczby węzłów, sieć stanowiąca taki element zawsze jest uwzględniona). Należy jeszcze raz zaznaczyć, że parametry kontrolne proponowanego rozwiązania nie były dostosowywane do zadań, podobnie jak w przypadku komitetu i *bagging*. W przypadku wzmacniania gradientowego natomiast, jak również w przypadku sieci neuronowej, porównano wyniki z użyciem kilku wartości parametru kontrolnego i wybrano najlepsze.

5.5.3. Porównanie wyników eksperymentów z wynikami w literaturze

Dzięki obecności użytych do testowania zadań w literaturze można porównać proponowane rozwiązanie z kilkoma innymi, nie przetestowanymi bezpośrednio.

Inne wersje proponowanego rozwiązania były w przeszłości [23, 24] testowane na zbiorach *housing* i *autoMpg*. Wersja obecna osiąga wyraźnie lepsze rezultaty (o ok. 5-10% błędu bezwzględnego) od każdego z tych rozwiązań na tych zbiorach.

W pracy [73] uwzględniono 6 metod łączenia drzew regresji w złożone struktury: lasy obrotów (*rotation forests* - proponowany we wspomnianym artykule), iterowany *bagging*, AdaBoost.R2-W, AdaBoost.R2-S i losowe podprzestrzenie. Procedura testowa była nieco inna - stosowana była 5-krotnie 2-krotna walidacja krzyżowa, co mogło dać gorsze wyniki. Wszystkie te rozwiązania zostały uruchomione na wszystkich zadaniach wspomnianych w niniejszej pracy i podano błąd RMSE. Na 15 z nich proponowane rozwiązanie uzyskało mniejszy błąd (RMSE) niż którykolwiek wśród 6 rozwiązań podanych w tamtym artykule, przy czym dla kilku (*housing, m_cpu, servo*) różnice były duże (dochodzące do 100% mniejszego błędu). Dla zadania δ *ailerons* wyniki były niemal identyczne. Na pięciu zbiorach przynajmniej jedno z 6

zaprezentowanych tam rozwiązań dało mniejsze błędy. Dla najlepszego pojedynczego rozwiązania z cytowanego artykułu liczba zadań z mniejszym błędem wyniosła 3 (m. in. *california housing*).

Z kolei w nieco starszej pracy [90], w której wprowadzany był algorytm AdaBoost.RT (z drzewem M5 jako podstawowym rozwiązaniem), algorytm ten został porównany z AdaBoost.R2, metodą z [12], *Big Error Margin* (BEM), *bagging* i siecią neuronową. We wszystkich trzech zadaniach, które były raportowane zarówno w niniejszej dysertacji, jak i w cytowanej pracy (*boston*, *autoMpg*, *friedman*) proponowane w niniejszej dysertacji rozwiązanie dało mniejsze błędy od wszystkich wymienionych tam rozwiązań.

W artykule [84] zaproponowana została metoda uczenia ekstremalnego (*extreme learning*) dla sieci o funkcjach radialnych (*radial basis function*), nazwana NME2-ELM, na podstawie algorytmu I-ELM [84]. Oprócz wymienionych metod w porównaniu uczestniczyły jeszcze odmiany uczenia ekstremalnego EI-ELM i CI-ELM, a także zwykła regresja na wektorach nośnych (*support vector regression* SVR). Raportowano RMSE dla zmiennych wyjściowych przeskalowanych do przedziału [0,1]. Na wszystkich 7 wymienionych tam zbiorach (*abalone, auto, boston, california housing, δailerons, δelevators, m_cpu*) rozwiązanie proponowane w niniejszej pracy dało rezultaty lepsze od dowolnego z wymienionych w [84] rozwiązań, ale kosztem wielokrotnie większego czasu nauki, zwłaszcza dla dużych zbiorów. W przypadku czterech z wymienionych w cytowanym artykule rozwiązań jest to skutek założeń uczenia ekstremalnego, które ma charakteryzować się właśnie przede wszystkim krótkim czasem uczenia, nawet kosztem dokładności.

W pracy [36] można znaleźć testy na 2 zbiorach użytych również w niniejszej pracy: *boston* i *kin8nm*, dla hierarchicznych mieszanin ekspertów. Błąd HME, ME, drzew MARS3.6 oraz sieci neuronowej jest wyraźnie większy niż błąd proponowanego rozwiązania na obydwu zbiorach. Komitet selekcyjny - metoda prezentowana przez autorów cytowanej pracy ma mniejszy błąd na zbiorze *boston*.

Hierarchiczną aproksymację można porównywać też do nieco mniej znanych rozwiązań. Uczenie "spójne z doświadczeniem" jest tematyką pracy [74]. Oznacza to uczenie maszynowe uwzględniające pewną uprzednią wiedzę o problemie. Ponieważ jednak bazuje na wielu regresjach liniowych, błędy (średniokwadratowe) na 5 z 6 opisanych tam zbiorach są większe, mniejsze natomiast na jednym zbiorze (*m_cpu*).

5.6. Podsumowanie

Wyniki rozwiązania dla 21 zadań opisanych w tym rozdziale można okreslić jako dobre. Hierarchiczna aproksymacja daje błędy mniejsze niż dobierana do zadania sieć neuronowa oraz komitet i *bagging*. Daje zbliżone rezultaty do dopasowywanego do zadania (względem jednego parametru) wzmacniania gradientowego. Pozytywnie wypada też porównanie z wymienionymi metodami dostępnymi w literaturze - błędy były w większości mniejsze lub zbliżone do błędów tych metod.

Wartym uwagi jest fakt, że rozwiązanie nie miało modyfikowanych żadnych parametrów kontrolnych, a więc nie wymagało dodatkowej kontroli użytkownika na żadnym z zadań i używało standardowych metod przetwarzania danych.

6. Przewidywanie zużycia energii elektrycznej w Polsce

6.1. Motywacja

Energia elektryczna jest jednym z podstawowych nośników energii wykorzystywanych we współczesnej gospodarce. Ma ona jednak istotną wadę: w skali przemysłowej nie można jej magazynować bezpośrednio, a jedynie po przekształceniu na inne rodzaje energii. Także w formie przekształconej przechowuje się ją jedynie w stosunkowo niewielkich ilościach, zarówno w systemie (gdzie mogą tego dokonywać głównie elektrownie szczytowo-pompowe), jak i u użytkowników końcowych. Jej chwilowa produkcja musi zatem niemal idealnie odpowiadać aktualnemu zużyciu. Przy obecnie używanych mocach, względy ekonomiczne sugerują jej produkcję w dużych ośrodkach i przez urządzenia mające pewną "bezwładność" produkcji - np. w Elektrowni Opole czas przejścia od stanu zimnego, czyli stanu po postoju ponad 48 godzin, do uzyskania pełnej mocy wynosi ok. 6 godzin [65], a nawet ze stanu gorącego (postój poniżej 6 godzin) ok. 4.5 godziny. Stąd wynika duża potrzeba prognoz krótkoterminowych (z wyprzedzeniem od godziny do tygodnia) [59, 69]. Prognozy średnio i długoterminowe użyteczne są przede wszystkim w logistyce (np. dostawy surowców do elektrowni, planowanie) i długoterminowej grze rynkowej.

Opracowano wiele różnych technik służących do przewidywania zużycia energii elektrycznej w krótkim horyzoncie czasowym, przykłady można znaleźć w [14, 35, 61, 71] lub monografii [59]. Wśród użytych metod można znaleźć zarówno klasyczne metody statystyczne, np. regresję liniową, autoregresję, filtry Kalmana, jak i systemy regułowe, a także metody uczenia maszyn, takie jak sztuczne sieci neuronowe, regresję opartą na wektorach nośnych, systemy rozmyte i rozmyto-neuronalne, algorytmy genetyczne [49, 55, 57, 59]. Znajdują tu też zastosowanie transformacje zmieniające dziedzinę, np. falkowe [57, 91]. Temat wciąż jest dość popularny, ostatnio powstały np. artykuły [13, 49, 55, 57, 91].

6.1.1. Podstawowe wiadomości o rynku energii elektrycznej w Polsce

Hurtowy obrót energią elektryczną w Polsce odbywa się przede wszystkim poprzez Towarową Giełdę Energii (TGE S.A.) [4, 5]. W zakresie energii elektrycznej funkcjonują tam:

1. Rynek Terminowy Towarowy.
2. Rynek Praw Majątkowych.
3. Rynek Dnia Bieżącego.
4. Rynek Dnia Następnego.

W kontekście prognoz krótkoterminowych ze wspomnianych powyżej najbardziej interesujące są Rynek Dnia Bieżącego i Rynek Dnia Następnego.

Na Rynku Dnia Bieżącego zawierane są kontrakty godzinowe na dostawę 1 MWh w konkretnej godzinie doby. Na 1 dzień przed dostawą, w godzinach 11:30 do 14:30 można zawierać kontrakty na dowolną godzinę następnej doby. W dobie dostawy, od godziny 8 do 14:30 można zawierać kontrakty na niektóre godziny obecnego dnia: w godzinach 8 do 8:30 na godziny 12-24, w godzinach 8:30 do 9:30 na godziny 13-24, 9:30 do 10:30 na godziny 14-24 itd. Zawierane kontrakty podsumowywane są w tzw. Grafikach Pracy, określających saldo dostawy/odbioru energii elektrycznej w każdej godzinie. Transakcje zgłaszane są do Operatora Sieci Przesyłowych [4, 10].

Rynek Dnia Następnego ma część, która działa podobnie do Rynku Dnia Bieżącego, z tym, że kontrakty na wszystkie godziny zawierane są w godzinach 8-14:30 na dwa dni przed terminem dostawy i w godzinach 8-13:30 na 1 dzień przed terminem dostawy. Oprócz tego (w podobnych terminach) można zawierać 3 rodzaje kontraktów blokowych na dostawę energii elektrycznej: w każdej godzinie doby, w każdej z godzin zwiększonego zapotrzebowania (7-22) oraz poza nimi (22-24 i 0-7). Na obydwu tych rynkach rozliczeniem jest rzeczywista dostawa energii, nie są dopuszczane rozliczenia czysto finansowe [4].

Rynek Bilansujący jest inną odmianą obrotu energią. Jest to rynek techniczny, którego działanie służy utrzymywaniu równowagi pomiędzy popytem i podażą energii elektrycznej, o której wspomniano wyżej, a jego operatorem jest Operator Sieci Przesyłowej (dla energii elektrycznej – Polskie Sieci Elektroenergetyczne)[6].

Na tym rynku operator sieci przede wszystkim kontraktuje rezerwę operacyjną mocy, a także udziały w regulacji pierwotnej o czasie reakcji od sekund lub wtórnej, o czasie reakcji rzędu minut, polegające na instalacji w określonych elektrowniach odpowiednich układów regulacji pierwotnej lub wtórnej [9]. Ze względu m.in. na koszty utrzymywania rezerw (zwł. przeznaczonych do szybkiego użycia) i niemożność przechowania energii niewykorzystanej, użycie mechanizmów rynku bilansującego

jest kosztowne [6, 4].
Obrót energią elektryczną odbywać się może poza wymienionymi wyżej rynkami w formie kontraktów dwustronnych [5].

Krótkoterminowe prognozy zużycia energii elektrycznej dla całej sieci mogą być przydatne m.in. do:

1. Ustalania ofert dostarczenia energii elektrycznej przede wszystkim na Rynkach Dnia Bieżącego i Dnia Następnego.
2. Ustalania dokładnych potrzeb rezerw mocy o czasie rozruchu liczonym w godzinach przy znajomości stanu transakcji na rynkach handlowych.

6.2. Zadanie

W niniejszym rozdziale przedstawione zostanie rozwiązanie prognozujące zużycie energii elektrycznej, a konkretnie *profil zużycia*, czyli wartości zużycia energii elektrycznej w 24 kolejnych godzinach. Mając dane z godziny h_s i wcześniejszych (zob. punkt 6.3.1), należy przewidzieć zużycia w chwilach $h_s + 1, h_s + 2, \ldots h_s + 24$. To zadanie jest identyczne jak rozwiązywane w [14]. W owej pracy wskazano godziny startowe dające najlepsze rezultaty, co potwierdziły wstępne testy własne. Uwzględniając te fakty oraz czas pracy rynków krótkoterminowych energii (podrozdział 6.1.1), uznano, że najlepsze czasy do startowania prognoz to godziny 8 i 11. Prognozy uzyskiwane o tych godzinach są dokładniejsze niż prognozy z większości innych godzin, a także są dostępne zaraz na początku sesji rynków krótkoterminowych.

6.3. Rozwiązanie

Podobnie jak w [14], zaproponowany został system hybrydowy. Składa się on z dwóch głównych części:

1. Zespołu 24 hierarchicznych aproksymacji – każdej dla innej godziny docelowej (czyli także innego wyprzedzenia prognozy), używanego dla dni "zwykłych": dni roboczych oraz sobót i niedziel niebędących dodatkowymi dniami wolnymi od pracy.
2. Zestawu 33 (30 dla startu o 8) regresji liniowych - 22 używanych do przewidywania zużycia w czasie dodatkowych dni wolnych od pracy (jedna na każdą godzinę takiego dnia), a 11 (8 przy starcie prognozy o godzinie 8) do przewidywania zużycia od końca dodatkowego dnia wolnego do początku następnej prognozy.

Pierwszą godziną, na którą przewidywanie odbywało się za pomocą regresji liniowej, była godzina 3 w nocy dodatkowego dnia wolnego od pracy. Oznacza to, że działała ona już w dzień poprzedzający. Regresja ta uczyła się różnicy pomiędzy zużyciem w godzinie docelowej, a zużyciem w punkcie "sklejenia" – godzinie 2 w nocy. Ponieważ prawdziwe zużycie z tej godziny nie jest dostępne w czasie przewidywania, użyto uprzedniej predykcji.

Podział taki został zastosowany ze względu na duże różnice w przebiegach zużycia energii elektrycznej pomiędzy dodatkowymi dniami wolnymi od pracy a pozostałymi [14], a także duże błędy w godzinach bezpośrednio następujących po dodatkowych dniach wolnych od pracy w rozwiązaniu, w którym regresja liniowa używana była tylko dla dodatkowych dni wolnych od pracy. Użycie uprzednich prognoz jako danych wejściowych oznacza, że system był rekursywny.

6.3.1. Dane wejściowe

Danymi wejściowymi dla części używanej dla "zwykłych" dni były:

1. Zużycie energii elektrycznej na 1, 2, 24, 25, 26, 48, 49 i 50 godzin przed godziną na którą wykonywana jest predykcja (h_p). Jeżeli prawdziwe dane były niedostępne, używana była uprzednio wykonana predykcja dla odpowiedniej godziny. Dane te zostały tak znormalizowane, aby zakres 7500MW - 27500MW odpowiadał przedziałowi [0,1].
2. Zużycie energii elektrycznej na godzinę, 25 godzin i 49 godzin przed godziną, **o** której wykonywana jest prognoza (8 lub 11).
3. Dane o temperaturze: prognozowana średnia temperatura z godziny, na którą wykonywana jest predykcja h_p i dwóch wcześniejszych $((T[h_p]+T[h_p-1]+T[h_p-2])/3)$ oraz analogiczne średnie z 24 $(T[h_p-24]+T[h_p-25]+T[h_p-26])/3)$ i 48 $(T[h_p-48]+T[h_p-49]+T[h_p-50])/3)$ godzin wcześniej. Temperatura została znormalizowana przez odjęcie 5 stopni i podzielenie rezultatu przez 20.
4. Dane o czasie:
   a) Dzień roku, znormalizowany tak, aby zakres zmiennej wyniósł $[-1, 1]$.
   b) Odległość danego dnia od środka roku, znormalizowana j.w.
   c) Zmienna binarna mówiąca, czy dzień jest niedzielą.
   d) Zmienna dyskretna mówiąca, czy dzień jest poniedziałkiem (-1), sobotą (+1), czy innym dniem (0).
   e) Dla pozostałych dni tygodnia – numer dnia tygodnia znormalizowany tak, jak w pkt. 1.
   f) Informacja czy poprzedzający dzień był dodatkowym dniem wolnym od pracy.

Dane użyte jako wejścia sieci neuronowych w pracy [14] miały podobną ogólną postać (wartości chwilowe i średnie z wcześniejszych zużyć energii elektrycznej oraz dane o temperaturze), ale różniły się szczegółami.

Dane przekazywane regresji liniowej:

1. Prognoza zużycia energii elektrycznej na godzinę "bazową" (2 w nocy) - tylko dla godzin od 3 do godziny startowania prognozy.
2. Zużycie energii w godzinie poprzedzającej godzinę, w której uruchamiana jest prognoza.
3. Temperatura z powyższej godziny.
4. Prognozowana temperatura z godziny, na którą dokonywana jest predykcja.
5. Czy dzień, na który wykonujemy prognozę, to dzień wolny, czy następujący po nim.
6. Zestaw zmiennych binarnych określających rodzaj święta.

Dane dotyczące energii elektrycznej zostały pobrane ze strony Polskich Sieci Elektroenergetycznych [6]. Oryginalne dane są danymi chwilowymi dla przedziałów 15-minutowych. Dla zmniejszenia czasu uczenia użyto tylko danych dotyczących pełnych godzin (lub średnich z danej godziny - w zależności od wersji zadania).

Prognozy dotyczące temperatury zostały wykonane w Interdyscyplinarnym Centrum Modelowania Uniwersytetu Warszawskiego, za pomocą modelu *Unified Model* (UM) przy pomocy oprogramowania MetOffice [2]. Wyniki tego modelu są obliczane czterokrotnie w ciągu doby dla warunków początkowych z godzin 0, 6, 12 i 18 (według czasu UTC) [3]. Dla prognozy zużycia energii elektrycznej uruchamianej o godzinie 11 użyto danych o temperaturze pochodzących z prognozy pogody z godziny 6, natomiast dla prognozy zużycia energii elektrycznej uruchamianej o 8 użyto danych pochodzących z prognozy pogody rozpoczynanej o północy. Są to najnowsze prognozy pogody, które mogą być użyte w danych godzinach, ze względu na ok. 4-godzinne oczekiwanie na wykonanie prognozy. Przewidywane temperatury obliczono dla miast wojewódzkich. Jako prognoza podawana była średnia prognozowana temperatura ważona liczbą ludności województw.

6.3.2. Parametry kontrolne

Regresja liniowa używała selekcji atrybutów analogicznej do drzew M5 [80]. W części używającej metody opisanej w niniejszej pracy zmieniono następujące parametry w stosunku do tych używanych w rozdziale 5: głębokość drzewa zmniejszono do 1, a współczynnik, z jakim karane są duże wagi (zob. punkt 4.3.3), został podniesiony do 0.005.

6.3.3. Procedura testowa

Rozwiązanie uczono najpierw na danych z lat 2009 do 2011, a testowano na roku 2012 (być może niezbyt szczęśliwie, ze względu na Mistrzostwa Europy w Piłce Nożnej 2012, które nie były uwzględnione w danych), co służyło przede wszystkim znajdowaniu parametrów kontrolnych, z wynikiem podanym powyżej. Następnie uczono rozwiązanie na danych z lat 2009-2012, a testowano na danych z pełnego roku 2013 (okresie 02-01-2013 do 01.01.2014 włącznie) i te wyniki są prezentowane w rozprawie. Test powtórzono 10-krotnie (jednak zmienność pomiędzy przebiegami była nieduża), a błąd uśredniono.

W tym eksperymencie raportowano przede wszystkim MAPE (*mean absolute percentage error*):

$$MAPE = \sum_{i=1}^{|S|} \frac{|y_i - g(x_i)|}{y_i} \tag{6.1}$$

wyrażony w procentach.

Testowano dwie godziny startowe dla prognoz: 8 i 11.

Testowano też dwie wersje zadania - przewidywanie wartości chwilowych zużycia energii elektrycznej albo przewidywanie średnich godzinowych (powstałych z uśrednienia po 4 wartościach z przedziałów 15-minutowych).

6.4. Wyniki i dyskusja

6.4.1. Przykładowy proces uczenia

Poniżej opisano proces tworzenia drzewa aproksymacji dla przewidywania średniego godzinowego zużycia energii elektrycznej na godzinę 20 przy starcie prognozy o godzinie 11. Drzewa aproksymacji dla innych godzin oraz dla innych wariantów były podobne.

Drzewo to, zgodnie z ustawionym parametrem kontrolnym, ma tylko dwa poziomy, co ogranicza czas uczenia. Na rysunku 6.1 podano w węzłach liczbę przykładów uczących oraz błąd na moduł (w MW) na zbiorze uczącym.

Tym razem wstępna selekcja cech przed klastrowaniem (pkt 1 algorytmu z podrozdziału 4.1, s. 43) wybrała jedynie 5 z 21 możliwych wektorów. Są one umieszczone w tablicy 6.1. Po grupowaniu i progowaniu rozmiary zbiorów uczących dzieci wyniosły 718, 835, 837, 602, 1148. Wielkości części wspólnych zbiorów zaznaczono w tablicy 6.2. Co ciekawe, pierwszy zbiór uczący został podzbiorem zbioru drugiego. Oba natomiast mają wiele wspólnych przykładów ze zbiorem piątym.

Tablica 6.1. Kierunki wybrane przez PCA przed klastrowaniem wyników w korzeniu przykładowego drzewa z godziny 20.

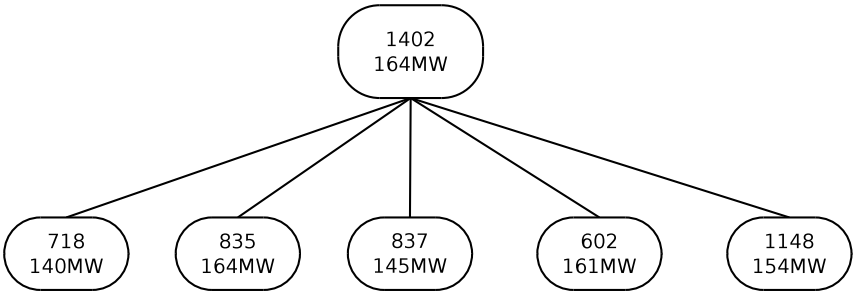
v_1	v_2	v_3	v_4	v_5
-0.1252	-0.0310	0.0594	-0.1425	-0.0282
0.0370	0.0064	-0.0007	-0.0470	-0.9275
-0.6472	-0.0548	0.0390	0.0805	-0.2477
-0.0233	0.9447	0.2537	-0.0345	0.0150
-0.0268	0.2227	-0.9397	-0.0275	0.0096
0.0188	0.0210	-0.0111	0.9067	-0.0710
0.0013	-0.0053	0.0151	0.0089	-0.0129
0.5467	0.0476	-0.0265	-0.0886	-0.2136
0.2705	0.0257	-0.0155	-0.0457	-0.1107
0.1775	0.0160	-0.0121	-0.0242	-0.0751
-0.1358	-0.0297	0.0595	-0.1542	-0.0272
-0.1306	-0.0297	0.0611	-0.1700	-0.0375
-0.1288	0.0148	-0.0733	-0.0882	-0.0282
-0.1393	0.0163	-0.0801	-0.0848	-0.0280
-0.1344	0.0186	-0.0889	-0.0861	-0.0393
-0.1254	0.0931	-0.0311	0.0334	-0.0292
-0.1357	0.0972	-0.0281	0.0396	-0.0280
-0.1307	0.1047	-0.0270	0.0449	-0.0387
-0.0774	-0.0241	0.0550	-0.2162	0.0141
-0.0802	0.0321	-0.1138	-0.0868	0.0032
-0.0750	0.1305	-0.0250	0.0489	0.0016

Tablica 6.2. Nakładanie się zbiorów uczących dla bezpośrednich potomków korzenia w przykładowym drzewie.

	c_1	c_2	c_3	c_4	c_5
c_1	718	718	212	91	607
c_2	718	835	323	140	724
c_3	212	323	837	470	735
c_4	91	140	470	602	434
c_5	607	724	735	434	1148

Wszystkie utworzone węzły potomne miały błąd na swoich zbiorach uczących mniejszy niż błąd korzenia i zostały zachowane. Każdy z nich był uczony tylko raz. Drugi węzeł potomny ma taki sam błąd na moduł na swoim zbiorze uczącym, co korzeń na swoim. Na zbiorze uczącym drugiego węzła potomnego średni błąd korzenia był jednak nieco większy i wyniósł 166MW. Ogólnie, dla tego zadania bardzo rzadko występowała potrzeba ponownego uczenia lub usuwania węzłów potomnych (pkt 4 głównego algorytmu, s. 22), co jest częściowo spowodowane po prostu mniejszą głębokością, ale do pewnego stopnia zapewne też charakterystyką zadania.

Jednokrotne uczenie całości systemu zajęło poniżej 2 minut na dwurdzeniowym procesorze komputera przenośnego. Odpowiedzi dla całego zbioru testowego (1 rok) uzyskiwane były w czasie poniżej sekundy.



Rysunek 6.1. Struktura przykładowego drzewa dla przewidywania energii elektrycznej z godziny 20. Liczby w węzłach: rozmiar zbioru uczącego (wyższa), błąd bezwzględny (niższa).

6.4.2. Skuteczność rozwiązania

W tablicach 6.3 i 6.4 przedstawiono wyniki dotyczące średniego błędu MAPE dla danego dnia tygodnia i godziny dla przewidywania średniej godzinowej, za godziny startu prognozy uznając odpowiednio 11 i 8. Tablice te nie uwzględniają specjalnych dni wolnych od pracy.

Można łatwo zauważyć oczekiwany fakt, że w "zwykłych" dniach tygodnia błędy są mniejsze. Jednak największe błędy występują nie dla niedzieli, która ma profil zużycia najbardziej odbiegający od profilów innych dni, ale w poniedziałek rano. Prognozy te jednak są wykonywane w niedziele, utrudnieniem w uczeniu może być więc rozbieżność między profilami zużycia dni, z których pochodzą dane wejściowe, a dniem docelowej prognozy. Ciekawym zjawiskiem jest istotnie, tzn. o kilkanaście procent, mniejszy błąd w środę niż we wtorek i czwartek. Większy błąd we wtorek może wynikać z użycia danych wejściowych z niedzieli, a więc mniej bezpośrednio użytecznych, z uwagi na różnice w profilach zużycia pomiędzy niedzielą a wtorkiem.

Można zauważyć ogólną zależność, że im dalej od startu prognozy, tym błędy większe, ale nie jest to zależność ściśle monotoniczna. Od godziny następującej po godzinie startu prognozy do godzin 17-20 błędy prognozy wzrastają. Nie jest to zwykłą pochodną występującego wtedy szczytu zapotrzebowania, ponieważ MAPE jest niezależny od skali.

Następnie błąd spada (ok. 0.2% MAPE) i osiąga ponowne minimum lokalne ok. godzin 21-23 (w zależności od dnia). Następnie rośnie i w godzinach 2-4 we wtorek, czwartek i piątek, o godzinie 6 w sobotę, środę i niedzielę oraz o 7 w poniedziałek, osiąga dla większości dni maksimum dobowe. Później, od godziny 7, w większości dni tygodnia trochę spada, choć jest to mniej zaznaczone w sobotę i niedzielę.

Co ciekawe, wiele z tych trendów jest niezależnych od czasu startu prognozy.

Tablica 6.3. Średni MAPE dla poszczególnych godzin doby i dni tygodnia, przy przewidywaniu średnich wartości godzinowych w dniach niebędących wolnymi od pracy. Godziną startu prognozy była 11.

Dzień Godzina	Nd	Pn	Wt	Śr	Czw	Pt	Sb	Średnio
0	0.84%	0.98%	0.88%	0.84%	0.72%	0.87%	0.88%	0.86%
1	0.98%	1.29%	1.33%	0.95%	0.89%	0.87%	0.97%	1.04%
2	1.09%	1.76%	1.71%	1.13%	1.34%	1.28%	1.03%	1.33%
3	1.23%	1.76%	1.54%	1.18%	1.56%	1.50%	1.23%	1.43%
4	1.51%	1.45%	1.51%	1.14%	1.60%	1.61%	1.41%	1.46%
5	1.77%	1.36%	1.49%	1.12%	1.49%	1.39%	1.43%	1.44%
6	1.89%	1.56%	1.26%	1.21%	1.18%	1.30%	1.49%	1.42%
7	1.55%	1.92%	1.04%	0.99%	0.94%	1.13%	1.34%	1.27%
8	1.47%	1.84%	0.90%	0.92%	0.91%	0.97%	1.29%	1.19%
9	1.44%	1.66%	0.95%	0.88%	0.96%	0.87%	1.32%	1.16%
10	1.29%	1.57%	1.01%	0.88%	1.02%	0.88%	1.24%	1.13%
11	1.22%	1.49%	1.09%	0.95%	1.11%	0.88%	1.12%	1.12%
12	0.40%	0.29%	0.25%	0.34%	0.27%	0.25%	0.33%	0.31%
13	0.57%	0.47%	0.36%	0.48%	0.41%	0.47%	0.56%	0.47%
14	0.71%	0.66%	0.60%	0.48%	0.61%	0.58%	0.73%	0.63%
15	0.90%	0.72%	0.62%	0.53%	0.75%	0.61%	0.86%	0.71%
16	0.91%	0.87%	0.64%	0.58%	0.80%	0.71%	0.90%	0.77%
17	0.93%	0.97%	0.79%	0.57%	0.83%	0.84%	0.88%	0.83%
18	1.05%	0.91%	0.74%	0.59%	0.76%	0.90%	0.93%	0.84%
19	0.98%	0.87%	0.70%	0.61%	0.69%	0.88%	0.80%	0.79%
20	0.86%	0.89%	0.75%	0.65%	0.56%	0.74%	0.87%	0.76%
21	0.80%	0.66%	0.69%	0.62%	0.50%	0.64%	0.83%	0.68%
22	0.74%	0.52%	0.62%	0.58%	0.63%	0.80%	0.79%	0.67%
23	0.77%	0.75%	0.75%	0.56%	0.70%	0.83%	0.80%	0.74%
Średnio	1.08%	1.13%	0.92%	0.78%	0.89%	0.90%	1.00%	**0.96%**

Pozwala to przypuszczać, że na ten błąd wpływ mają czynniki zewnętrzne, zwiększające chaotyczność zużycia energii elektrycznej w określonych porach dnia lub dniach tygodnia, ewentualnie zmienne nie uwzględnione w modelu, których wpływ zwiększa się w tych godzinach.

Ponieważ błędy te pojawiają się w okolicach zwiększonej aktywności bytowej, nie zaś zawodowej – mniej więcej w czasie wychodzenia z domu i tuż po powrocie z pracy dużej liczby ludzi, także w weekendy, kiedy nie tylko błąd jest trochę większy przez cały dzień, ale jego poranny szczyt także pojawia się później – można przypuszczać, że z nią właśnie mają związek. Czynnikiem może być tu np. nasłonecznienie, powodujące włączanie lub nie świateł czy ogrzewania, ale też fakt, czy w danym momencie telewizja emituje wyjątkowo ciekawy program. Niestety, takie dane nie były dostępne w czasie powstawania niniejszej pracy.

W tablicy 6.5 zamieszczono wyniki dla dni dodatkowo wolnych od pracy, uzyskane przy pomocy zestawu regresji liniowych. Elementy charakterystyczne dla prze-

Tablica 6.4. Średni MAPE dla poszczególnych godzin doby i dni tygodnia, przy przewidywaniu średnich wartości godzinowych w dniach niebędących wolnymi od pracy. Godziną startu prognozy była 8.

Dzień Godzina	Nd	Pn	Wt	Śr	Czw	Pt	Sb	Średnio
0	0.94%	0.97%	0.93%	0.85%	0.82%	0.93%	0.89%	0.90%
1	0.98%	1.32%	1.31%	0.90%	0.91%	0.97%	0.96%	1.05%
2	1.13%	1.83%	1.76%	1.14%	1.24%	1.30%	1.00%	1.34%
3	1.28%	1.89%	1.66%	1.23%	1.43%	1.60%	1.17%	1.46%
4	1.55%	1.66%	1.56%	1.22%	1.45%	1.73%	1.40%	1.51%
5	1.93%	1.75%	1.56%	1.28%	1.42%	1.48%	1.47%	1.56%
6	2.08%	2.17%	1.30%	1.40%	1.30%	1.23%	1.54%	1.58%
7	1.77%	2.20%	1.09%	1.15%	1.05%	1.05%	1.49%	1.41%
8	1.56%	2.00%	0.99%	0.93%	0.99%	1.02%	1.38%	1.27%
9	0.71%	0.42%	0.40%	0.39%	0.37%	0.36%	0.59%	0.46%
10	1.02%	0.66%	0.69%	0.56%	0.67%	0.66%	0.84%	0.73%
11	1.22%	0.81%	0.89%	0.76%	0.85%	0.73%	0.95%	0.89%
12	1.25%	0.85%	0.91%	0.78%	0.96%	0.79%	1.06%	0.94%
13	1.26%	0.91%	0.94%	0.74%	1.01%	0.81%	1.19%	0.98%
14	1.36%	0.99%	1.03%	0.75%	1.14%	0.82%	1.19%	1.04%
15	1.46%	1.10%	1.02%	0.80%	1.24%	0.84%	1.17%	1.09%
16	1.48%	1.19%	0.99%	0.87%	1.28%	0.92%	1.16%	1.13%
17	1.46%	1.26%	1.00%	0.82%	1.19%	1.02%	1.17%	1.13%
18	1.41%	1.10%	0.92%	0.82%	0.95%	1.00%	1.13%	1.05%
19	1.33%	1.00%	0.85%	0.81%	0.84%	0.93%	0.96%	0.96%
20	1.22%	0.97%	0.85%	0.75%	0.74%	0.79%	1.05%	0.91%
21	1.01%	0.85%	0.80%	0.72%	0.70%	0.69%	0.94%	0.82%
22	0.81%	0.63%	0.70%	0.70%	0.79%	0.78%	0.88%	0.76%
23	0.78%	0.79%	0.72%	0.68%	0.78%	0.87%	0.84%	0.78%
Średnio	1.29%	1.22%	1.03%	0.88%	1.01%	0.97%	1.10%	**1.07%**

biegu błędu omówione dla "zwykłych" dni również tutaj się zaznaczają, choć z pewnymi modyfikacjami. Najbardziej widoczną z tych zmian jest znaczące zwiększenie błędów, ale znając pracę [14], nie można było się spodziewać niczego innego. Popołudniowe "dobre" przewidywania są tutaj trochę późniejsze, późne są także wysokie błędy. Co ciekawe, dużo większe błędy wykazuje aproksymacja okresu tuż po święcie, gdzie – zwłaszcza o godzinie 7 – błędy są bardzo duże. Przynajmniej częściowo za ten stan wydaje się odpowiadać rozbieżność między "zwykłymi" dniami po dniach wolnych i "długimi weekendami", w których przebieg wyraźnie się różni, a których obecnie używany zestaw regresji liniowych nie potrafi dobrze odróżnić. Właśnie system przewidujący na "dzień po" specjalnym święcie wydaje się mieć najwięcej potencjału do zmniejszenia błędu. Podobne, choć nieco bardziej ogólne, spostrzeżenie zostało umieszczone w [14].

W tablicy 6.6 można znaleźć podsumowanie błędów rozwiązania dla zbiorów uczących i testujących z różnymi godzinami startowymi i dokładnym celem przewi-

Tablica 6.5. Średni MAPE przy przewidywaniu średniego zużycia godzinowego, dla poszczególnych godzin doby dla okresów "specjalnych" - dodatkowych dni wolnych od pracy i okresów bezpośrednio po nich. Godziną startu prognozy była 11:00.

Dni dodatkowe		Dni następujące po dodatkowych	
Godzina	MAPE	Godzina	MAPE
3	2.04 %	1	2.23%
4	2.56 %	2	1.97%
5	3.23 %	3	4.05%
6	2.92 %	4	2.95%
7	3.54 %	5	2.76%
8	3.66 %	6	5.16%
9	4.30%	7	12.04%
10	4.24%	8	4.56%
11	5.06%	9	8.40%
12	1.61%	10	5.66%
13	1.43%	11	7.86%
14	1.47%		
15	1.81%		
16	2.29%		
17	2.09%		
18	2.01%		
19	2.25%		
20	2.06%		
21	1.91%		
22	1.66%		
23	1.91%		
24	2.29%		
Łącznie 3:00-24:00	2.51%	Łącznie 1:00-11:00	4.98%

dywań (średnia z 4 wartości czy wartość chwilowa). Przewidywanie średniej z danej godziny jest obarczone mniejszym błędem, gdyż krótkotrwałe fluktuacje uwzględniane w danych chwilowych są przez uśrednianie wygładzane, nawet jeśli ta średnia brana jest tylko z 4 wartości odległych o 15 minut.

Tablica 6.6. Podsumowanie błędów rozwiązania dla średniego zużycia godzinowego oraz chwilowego zużycia o pełnych godzinach. Dla całego roku, łącznie ze świętami dodatkowymi.

Start	średnia/chwilowa	Zbiory testujące		Zbiory uczące	
		MAPE	E_{abs}	MAPE	E_{abs}
11	średnia	1.08 %	186 MW	0.95%	160 MW
8	średnia	1.16 %	200 MW	1.00%	170 MW
11	chwilowa	1.15 %	199 MW	1.01%	171 MW
8	chwilowa	1.24 %	213 MW	1.06%	180 MW

Ponadto, w zgodzie z obserwacjami z [14], godzina 11 jest lepszą godziną do startowania prognoz, choć prognozy średnich godzinowych z godziny 8 też należy uznać za dokładne. Biorąc pod uwagę, że opisywane zjawisko może być wyraźnie zmienne

w czasie oraz fakt, że testowano rozwiązanie na danych z całego nowego roku bez douczania, różnice błędów pomiędzy zbiorami testującymi i treningowymi nie są duże.

6.4.3. Porównanie z metodami dostępnymi w literaturze

Biorąc pod uwagę wyniki opisywane w innych pracach dla zadań krótkoterminowego przewidywania zużycia energii elektrycznej, wynik MAPE 0.96% w dniach "zwykłych", a 1.08% po uwzględnieniu dni specjalnych, wypada dobrze. W pracy [14], w której metodyka była najbardziej podobna (i po części posłużyła za podstawę niniejszej), najlepszy raportowany MAPE wynosił 1.1% dla systemu hybrydowego opierającego się na podobnych zasadach, co system proponowany, jednak różnego w istotnych szczegółach. W systemie przewidującym zużycie w dniach dodatkowo wolnych od pracy użyto systemu eksperckiego bazującego na średnich zużyciach z poszczególnych godzin. Jako podstawowych rozwiązań dla dni "zwykłych" użyto w cytowanej pracy sieci neuronowych. Dane we wspomnianej pracy były również najbardziej podobne, choć dane o temperaturze pochodziły z innego źródła, a całość odnosiła się do innego (wcześniejszego o 10 lat) okresu.

Podanego ogólnego problemu – krótkoterminowego przewidywania zużycia energii elektrycznej – dotyczyło też wiele prac z ostaniach lat, chociaż konkretne problemy – obszar, z którego przewidywano i dokładny przebieg prognozy – różniły się. Na przykład w pracy [55], w której dla każdej godziny osobno używano systemu sieci połączonych algorytmem "bagging", MAPE wyniósł ok. 1.75%. Zużycie zostało zarejestrowane w Nowej Anglii. W zasadzie przewidywanie było zawsze z horyzontem 24-godzinnym, ale z powodu użycia średniej zużycia z poprzedniego dnia, stawało się przewidywaniem profilu (z godziną startową 00:00).

Podobnie zostało skonstruowane przewidywanie dla pracy [57], w której rozwiązanie było testowane na danych z 4 miesięcy. Osiągnięto tam MAPE 2.00% przy użyciu prognozy temperatury, nieco mniej przy użyciu rzeczywistej temperatury. Dla zużycia energii elektrycznej na obszarze Nowego Jorku przy użyciu do przewidywania transformat falkowych dla okresu testowego wynoszącego jeden miesiąc (lipiec) zarejestrowano błąd MAPE 1.82%.

Z kolei w pracy [49] również dla Nowego Jorku, ale dla okresu testowego 3 miesięcy, MAPE wyniósł 3.34%. W artykule tym do jednoczesnego znajdowania dobrych cech i optymalizacji parametrów regresji wektorów nośnych (SVR) używany jest algorytm memetyczny (*memetic algorithm*) oparty na metodach optymalizacji rojem cząstek (*particle swarm optimization*).

Niższe błędy raportowano dla innych danych w pracy [91], używającej transformat falkowych połączonych z addytywnym modelem Holta-Wintersa oraz metodą ważonych najbliższych sąsiadów. Osiągnięto tam MAPE 1.02 % i 1.09% dla zużycia odpowiednio w Kalifornii i Hiszpanii. Okresy testowe były jednak w tej pracy krótkie - wynosiły odpowiednio 3 i 4 tygodnie, każdy wybrany z innej pory roku.

Uzyskiwane w niniejszej pracy błędy leżą w zasięgu raportowanych przez rozwiązania komercyjne, np. TESLA, w którym z użyciem większej różnorodności danych wejściowych raportowane MAPE wynoszą 0.84-1.56% [7], jednak dotyczą przewidywań z użyciem rzeczywistej temperatury. Błąd MAPE z użyciem temperatury prognozowanej jest tam szacowany jako potencjalnie wyższy o 0.5 - 1% MAPE, z kolei uwzględnianie ostatnich trendów w profilu zużycia ma podnosić skuteczność o 0.15 – 0.30%. Po zsumowaniu dostajemy zakres 1.04% – 2.41%.

Warto tu dodać, że zmiana rozwiązania wobec tych użytych dla zupełnie odmiennych zadań w rozdziale 5 była dość prosta i wynikała przede wszystkim po prostu z chęci skrócenia czasu uczenia. Wstępne wyniki z rozwiązaniem identycznym do tego z rozdziału 5 były o 0.04% MAPE wyższe od opisywanego (MAPE ok. 1.12%), ale nie zostały ukończone ze względu na większą czasochłonność i sygnalizowaną nieco gorszą generalizację. Z drugiej strony – fragmentaryczne testy z użyciem sieci neuronowej z dobraną odpowiednią liczbą neuronów dawały podobne (MAPE ok. 1.15% dla przewidywania średniej startującego o godzinie 11) rezultaty.

6.5. Podsumowanie

W niniejszym rozdziale oszacowano przydatność proponowanego rozwiązania do tworzenia systemu przewidującego profil zużycia energii elektrycznej na następne 24 godziny z rozdzielczością godzinową (używaną też np. na Rynku Bilansującym w Polsce [6]). Oszacowanie to wypadło pozytywnie, wyniki są zbliżone lub lepsze od wyników podobnych zadań dostępnych w literaturze oraz zbliżone lub nieco lepsze od wyników rozwiązania komercyjnego.

Ponadto wskazano możliwe obszary oraz sposoby rozwoju i polepszenia skuteczności rozwiązania w postaci dołączenia dodatkowych danych, np. o nasłonecznieniu lub ew. dodatkowych specjalnych okazjach zwiększających zużycie prądu (dotyczących np. telewizji), znalezienia lepszego modelu do przewidywania zużycia w pierwszej połowie dnia po dodatkowym dniu wolnym od pracy oraz uwzględnienia “długiego weekendu”.

7. Podsumowanie i wnioski

W niniejszej dysertacji opisano metodę łączenia prostych rozwiązań dotyczących przewidywania wartości funkcji o przeciwdziedzinie rzeczywistej wielowymiarowej w jedno rozwiązanie o potencjalnie mniejszym błędzie. Metoda ta opiera się na podziale problemu na podproblemy z wykorzystaniem zmiennych wejściowych i odpowiedzi oraz błędów niektórych z uprzednio utworzonych rozwiązań jednostkowych. Podproblemy oraz rozwiązania zorganizowane są w strukturę drzewa. Każde z rozwiązań uczone jest dla pewnej części problemu (cały problem w korzeniu). Podproblemy nakładają się, a w czasie działania dla każdego przykładu używane są odpowiedzi kilku rozwiązań jednostkowych, ale zasadniczo nie wszystkich. Odpowiedzi te są uzyskiwane zarówno w liściach, jak i węzłach wewnętrznych.

W rozprawie opisano ogólny schemat algorytmu (zob. rozdz. 2, opisany wcześniej przez autora niniejszej pracy w [24]) oraz przedstawiono jego uszczegółowienia: sposoby rozwiązania podzadań tworzenia rozwiązań składowych w węźle drzewa oraz podziału zbioru uczącego i tworzenia przestrzeni kompetencji dla węzłów potomnych (zob. rozdz. 4). Oszacowano też złożoność czasową algorytmu z uwzględnieniem uszczegółowień (zob. podrozdział 4.4).

Ponadto, udowodniono twierdzenia dotyczące prezentowanej metody (zob. rozdz. 3, dowody w dodatku A). Najważniejsze z nich wyprowadzają błąd średniokwadratowy poddrzewa rozwiązania zaczepionego w danym węźle na danym zbiorze w zależności od:

1. Błędów średniokwadratowych poddrzew zaczepionych w węzłach potomnych i błędu rozwiązania obecnego w danym węźle *na ich zbiorach kompetencji*, które nakładają się, ale nie są tożsame.
2. Składnika jakości funkcji kompetencji – relatywnego do funkcji, która wszystkim wybranym węzłom przydziela równą kompetencję (a zatem można łatwo sprawić, by składnik ten wyniósł 0).
3. Dodatkowego składnika związanego z nierównoległością funkcji kompetencji i odchyleń od poprawnego wyniku odpowiedzi węzłów. Składnik ten jest zawsze niedodatni, a ściśle ujemny zawsze poza bardzo trudnym do uzyskania przypadkiem (zob. podrozdział 3.4). Zatem zmniejsza on błąd.

Twierdzenia te, w połączeniu z twierdzeniem z podrozdziału 3.5, pozwalają stwierdzić, że błąd średniokwadratowy na zbiorze uczącym danego węzła wraz z jego niepustym poddrzewem jest prawie zawsze mniejszy od błędu samego rozwiązania obecnego w węźle drzewa (bez uwzględniania poddrzew). Dla zbioru testującego wyznaczają one główne składniki błędu.

Wyniki teoretyczne posłużyły również jako podstawa dla rozwiązań szczegółowych konkretyzujących schemat algorytmu opisanych w rozdziale 4.

W niniejszej rozprawie prezentowane są także eksperymenty dotyczące skuteczności rozwiązania. Wyniki proponowanej metody w 21 ogólnie dostępnych zadaniach uczenia maszynowego porównano z wynikami sieci neuronowych i trzech innych metod łączenia aproksymacji funkcji w jedną dokładniejszą (rozdz. 5). Na podstawie proponowanej metody wykonano też system przewidujący profil zużycia energii elektrycznej w Polsce (rozdz. 6).

Z wniosków z twierdzeń, głównie z twierdzenia 1, s. 35 i podrozdziału 3.5 oraz eksperymentów można wysnuć wniosek, że opisywany sposób łączenia rozwiązań składowych rzeczywiście pozwala uzyskać mniejszy błąd, także poza zbiorem uczącym. Z eksperymentu porównawczego i porównania wyników z literaturą wynika, że dla wielu zadań rozwiązanie wykazuje błędy niższe od błędów dużej części innych rozwiązań stosowanych dla tego rodzaju zadań, w szczególności dla zadania przewidywania profilu zużycia energii. Ani w eksperymentach, ani w przytaczanej literaturze nie znaleziono też żadnej metody, która dawałaby istotnie lepsze sumaryczne wyniki od proponowanego rozwiązania. Oznacza to *dobre wyniki* proponowanej metody.

Również na podstawie eksperymentu porównawczego, w którym dla wszystkich 21 zadań parametry kontrolne były takie same oraz eksperymentu dotyczącego elektroenergetyki wynika, iż rozwiązanie daje dobre wyniki nawet z domyślnymi wartościami parametrów kontrolnych dla szerokiego zakresu zadań.

Powyższe pozwala stwierdzić, że teza pracy została wykazana, a cel osiągnięty.

Rozwiązanie nie jest, oczywiście, bez wad. Jego błędy, uśrednione po wielu zadaniach, nie są niższe od analogicznych średnich dla najlepszych stosowanych rozwiązań, takich jak wzmacnianie gradientowe (zob. rozdz. 5). Zastosowane szczegółowe rozwiązania, takie jak analiza składowych głównych czy sieci neuronowe, są zależne od przetwarzania danych wejściowych, choć w eksperymentach opisanych w rozdziale 5 użyto standardowych technik. Co więcej, czasy uczenia są dość duże (zob. tab. B.8 i podrozdział 4.4), zwiększane łatwo przez zwiększenie głębokości drzewa (które jednak można ograniczyć do pożądanego rozmiaru), a także przez zwiększenie liczby wymiarów, choć temu ostatniemu można przynajmniej częściowo

zaradzić stosując inne rozwiązania szczegółowe. W przypadku zmiennej wyjściowej wielowymiarowej zmniejszenie czasu uczenia można uzyskać poprzez rozpatrywanie każdej współrzędnej wyjścia oddzielnie, choć potencjalnie może to zwiększyć błąd. Rozwiązanie jednak można stosować dla wyjść będących wektorami rzeczywistymi, choć należałoby zmienić algorytm uczenia, na zależny np. liniowo od liczby wag sieci neuronowej (inny niż użyte odmiany wstecznej propagacji), ewentualnie zmienić architekturę sieci składowych. W tym względzie istotne jest jednak też to, że złożoność ze względu na liczbę przykładów uczących dla danych o umiarkowanej liczbie wymiarów jest liniowa.

Decyzje dotyczące rozwiązania wynikały w dużej mierze z chęci ograniczenia nadmiernego dopasowania. Taką rolę spełnia tu np. brak modyfikacji funkcji celu, przetwarzania zmiennych wejściowych, czy nawet brak podawania na wejście węzłom potomnym wyników rodziców (co było testowane we wstępnych eksperymentach i dawało mniejsze błędy na zbiorach uczących, większe na testujących). W tym świetle należy także rozpatrywać niewielkie (2-6) liczby węzłów ukrytych w sieciach neuronowych, składnik regularyzujący (zob. podrozdział 4.3.3 str. 48), ograniczanie liczby wag przez liczbę przykładów (podrozdział 4.2) podzieloną przez ustaloną liczbę czy rezygnację z tworzenia węzłów dla zbyt małych liczb przykładów (tamże). Ogólnie, połączenie tych technik można uznać za skuteczne, różnice w NRMSE pomiędzy zbiorami testowymi a uczącymi (tab. 5.5) są mniejsze niż we wzmacnianiu gradientowym, choć większe niż w przypadku zwykłych komitetów.

Niewielkie wymogi dotyczące ustawiania parametrów kontrolnych są wynikiem przede wszystkim właśnie wspomnianej wewnętrznej kontroli złożoności – choć parametrów możliwych do ustawienia jest dużo, wartości domyślne zachowują się dobrze, a głównym parametrem, który można zmienić, jest czas nauki – poprzez głębokość drzewa (możliwe są również inne rodzaje tej regulacji). Kontrola ta nie wykorzystuje danych walidujących, co zmniejsza wymagania dotyczące liczby przykładów uczących, ale jednak pozostawia ocenę generalizacji w sferze zgrubnych oszacowań, co może zwiększać błąd.

Dlatego kontrola złożoności i generalizacji z wykorzystaniem automatycznej walidacji może być jednym z kierunków rozwoju rozwiązania. Należy też zauważyć, że wagi przykładów są znane już w czasie uczenia, nie mają na nie wpływu wyniki nauki poszczególnych węzłów, jak w wielu innych rozwiązaniach [37, 67]. Wstępne eksperymenty dotyczące wersji proponowanego rozwiązania modyfikującej funkcję kompetencji ze względu na wyniki nauki bezpośrednich potomków danego węzła wykazały wysokie błędy na zbiorach testowych, ale może to wynikać właśnie z tego, że używane były błędy na zbiorach uczących, a nie pewnego rodzaju walidacji.

Nadal poszukiwane mogą być nowe metody podziału zbiorów uczących i uzyskiwania funkcji kompetencji. Być może dałoby się tu użyć reprezentacji pośrednich z rozwiązań, np. aktywacji węzłów ukrytych w sieciach, jako podstawy podziału. Mogłoby to dać dobre rezultaty, ponieważ aktywacje takie są wynikiem działania transformacji nieliniowej, a więc pewnej zmiany reprezentacji, która powstała w odpowiedzi na dostosowywanie sieci do zadania, będącej zatem potencjalnie bardziej użyteczną w rozróżnianiu "prawdziwych" podziałów. Oznaczałoby to jednak większe uzależnienie się od postaci rozwiązań składowych obecnych w węzłach drzewa rozwiązania.

Wspomniane reprezentacje pośrednie można też wykorzystać do uczenia aproksymacji w węzłach potomnych danego węzła, jako ułatwienie, a jednocześnie wytworzenie głębszej struktury. Użycie *wyjść* rodzica jako wejść węzłów potomnych we wstępnych eksperymentach nie dało jednak dobrych rezultatów. Można też poszukiwać innych algorytmów uczenia, np. takich, które przynajmniej częściowo zneutralizują wady rozwiązania. Wysoki koszt obliczeniowy mogą pomóc zmniejszyć rozwijające się od pewnego czasu metody uczenia ekstremalnego [50].

A. Dowody

Aby pomniejszyć objętość kolejnych dowodów, poniżej zostaną podane proste, ale bardzo przydatne tożsamości:

$$
\begin{aligned}
\widetilde{\eta}_i(x_k)_l =& \widetilde{g}_i(x_k)_l - y_{kl} \\
& \text{z definicji odpowiedzi węzła (def. 2)} \\
=& \sum_{j=0}^{N(i)} \widetilde{g}_{I(i,j)}(x_k)_l \cdot C_i(j, x_k) - y_{kl} \\
& \text{z definicji sumowania się funkcji kompetencji do 1 (równ. 2.5)} \\
=& \sum_{j=0}^{N(i)} \widetilde{g}_{I(i,j)}(x_k)_l \cdot C_i(j, x_k) - \sum_{j=0}^{N(i)} C_i(j, x_k) \cdot y_{kl} \\
& \text{ponieważ w obu sumach występują te same wagi i z definicji } \tilde{\eta} \text{ (równ. 3.3)} \\
=& \sum_{j=0}^{N(i)} (\widetilde{g}_{I(i,j)}(x_k)_l - y_{kl}) \cdot C_i(j, x_k) = \sum_{j=0}^{N(i)} \widetilde{\eta}_{I(i,j)}(x_k)_l \cdot C_i(j, x_k) \qquad \text{(A.1)} \\
& \text{wyrazy, w których } C_i(j, x_k) = 0 \text{, nic nie wnoszą do sumy, więc} \\
=& \sum_{j: C_i(j,x_k) > 0} \widetilde{\eta}_{I(i,j)}(x_k)_l \cdot C_i(j, x_k) \qquad \text{(A.2)}
\end{aligned}
$$

Dowody twierdzeń z podrozdziału 3.2 (o pojedynczym przykładzie)

Niech $\sum_{j: C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 > 0$. Z równiania (A.1) i definicji błędu kwadratowego 3.1:

$$
\begin{aligned}
e(\widetilde{g}_i, x_k) &= \sum_{l=1}^{r} \left(\widetilde{\eta}_i(x_k)_l\right)^2 = \sum_{l=1}^{r} \left(\sum_{j=0}^{N(i)} (\widetilde{\eta}_{I(i,j)}(x_k)_l) \cdot C_i(j, x_k) \right)^2 \\
&= \sum_{l=1}^{r} \left(\sum_{j=0}^{N(i)} (\widetilde{\eta}_{I(i,j)}(x_k)_l) \cdot \sqrt{C_i(j, x_k)} \cdot \sqrt{C_i(j, x_k)} \right)^2
\end{aligned}
$$

Z własności iloczynu skalarnego

$$e(\widetilde{g}_i, x_k) = \sum_{l=1}^{r} \left(\sqrt{\sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot \sqrt{C_i(j,x_k)}^2} \cdot \sqrt{\sum_{j=0}^{N(i)} \sqrt{C_i(j,x_k)}^2} \cdot \right.$$
$$\left. \cdot \cos\left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)}, \left[\sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)} \right) \right)^2$$
$$= \sum_{l=1}^{r} \left| \sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot \left| C_i(j,x_k) \right| \right| \cdot \left| \sum_{j=0}^{N(i)} \left| C_i(j,x_k) \right| \right| \cdot$$
$$\cdot \cos^2\left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)}, \left[\sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)} \right)$$

Wszystkie wyrazy pod modułami są nieujemne, a zatem

$$e(\widetilde{g}_i, x_k) = \sum_{l=1}^{r} \sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot C_i(j,x_k) \cdot \sum_{j=0}^{N(i)} C_i(j,x_k) \cdot$$
$$\cdot \cos^2\left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)}, \left[\sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)} \right)$$

suma funkcji kompetencnji to 1

$$= \sum_{l=1}^{r} \sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot C_i(j,x_k) \cdot$$
$$\cdot \cos^2\left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)}, \left[\sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)} \right)$$

Co dowodzi równości (3.6). □

Druga równość (3.7) wynika z dwóch prostych obserwacji. Po pierwsze, wyrazy $\tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot C_i(j,x_k)$ z $C_i(j,x_k) = 0$ są zerowe, więc można je opuścić w sumowaniu:

$$\sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot C_i(j,x_k) = \sum_{j: C_i(j,x_k) > 0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot C_i(j,x_k)$$

Po drugie, z własności cosinusa, standardowego iloczynu skalarnego dla liczb rzeczywistych i funkcji kompetencji (nieujemność), można doprowadzić odpowiednie

wyrażenia do analogicznej postaci - wyłączamy z sumy wyrazy zerowe:

$$\cos\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l\cdot\sqrt{C_i(j,x_k)}\right]_{j=0}^{N(i)},\left[\sqrt{C_i(j,x_k)}\right]_{j=0}^{N(i)}\right)=$$
$$=\frac{\sum_{j=0}^{N(i)}\tilde{\eta}_{I(i,j)}(x_k)_l\cdot C_i(j,x_k)}{\sqrt{\sum_{j=0}^{N(i)}\left(\tilde{\eta}_{I(i,j)}(x_k)_l\cdot\sqrt{C_i(j,x_k)}\right)^2}\cdot\sqrt{\sum_{j=0}^{N(i)}C_i(j,x_k)}}$$
$$=\frac{\sum_{j:C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l\cdot C_i(j,x_k)}{\sqrt{\sum_{j:C_i(j,x_k)>0}\left(\tilde{\eta}_{I(i,j)}(x_k)_l\cdot\sqrt{C_i(j,x_k)}\right)^2}\cdot\sqrt{\sum_{j:C_i(j,x_k)>0}C_i(j,x_k)}}$$
$$=\cos\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l\cdot\sqrt{C_i(j,x_k)}\right]_{j:C_i(j,x_k)>0},\left[\sqrt{C_i(j,x_k)}\right]_{j:C_i(j,x_k)>0}\right)$$

Wektory $\left[\tilde{\eta}_{I(i,j)}(x_k)_l\cdot\sqrt{C_i(j,x_k)}\right]_{j:C_i(j,x_k)>0}$ oraz $\left[\sqrt{C_i(j,x_k)}\right]_{j:C_i(j,x_k)>0}$ są rzutami pierwotnych wektorów na podprzestrzeń powstałą przez wyzerowanie odpowiednich współrzędnych, ale te współrzędne miały już wartość 0, stąd cosinus można liczyć także w podprzestrzeni.

Dowód lematu 2 przeprowadzany jest analogicznie do poprzedniego. Znów korzystając z tego, że $\sum_{j:C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2>0$, z równania (A.1) i własności błędu kwadratowego (3.5) wyprowadzamy:

$$e(\tilde{g}_i,x_k)=\sum_{l=1}^{r}(\tilde{\eta}_i(x_k)_l)^2=\sum_{l=1}^{r}\left(\sum_{j=0}^{N(i)}(\tilde{\eta}_{I(i,j)}(x_k)_l)\cdot C_i(j,x_k)\right)^2$$

z własności iloczynu skalarnego

$$=\sum_{l=1}^{r}\left(\sqrt{\sum_{j=0}^{N(i)}\tilde{\eta}_{I(i,j)}(x_k)_l^2}\cdot\sqrt{\sum_{j=0}^{N(i)}C_i(j,x_k)^2}\cdot\right.$$
$$\left.\cdot\cos\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l)\right]_{j=0}^{N(i)},[C_i(j,x_k)]_{j=0}^{N(i)}\right)\right)^2$$
$$=\sum_{l=1}^{r}\left|\sum_{j=0}^{N(i)}\tilde{\eta}_{I(i,j)}(x_k)_l^2\right|\cdot\left|\sum_{j=0}^{N(i)}C_i(j,x_k)^2\right|\cdot\left(\cos\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l)\right]_{j=0}^{N(i)},[C_i(j,x_k)]_{j=0}^{N(i)}\right)\right)^2$$

Wszystkie wyrazy pod modułami są $\geqslant 0$

$$=\sum_{l=1}^{r}\sum_{j=0}^{N(i)}\tilde{\eta}_{I(i,j)}(x_k)_l^2\cdot\sum_{j=0}^{N(i)}C_i(j,x_k)^2\cdot\left(\cos\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l)\right]_{j=0}^{N(i)},[C_i(j,x_k)]_{j=0}^{N(i)}\right)\right)^2$$

Całkowicie analogicznie można udowodnić drugą część, jedynie tym razem korzysta-

jąc od razu z równości (A.2) zamiast tylko z (A.1)

$$e(\widetilde{g}_i, x_k) = \sum_{l=1}^{r} (\tilde{\eta}_i(x_k)_l)^2 = \sum_{l=1}^{r} \left(\sum_{j:C_i(j,x_k)>0} (\tilde{\eta}_{I(i,j)}(x_k)_l) \cdot C_i(j, x_k) \right)^2$$

z własności iloczynu skalarnego

$$= \sum_{l=1}^{r} \left(\sqrt{\sum_{j:C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2} \cdot \sqrt{\sum_{j:C_i(j,x_k)>0} C_i(j, x_k)^2} \cdot \right.$$
$$\left. \cdot \cos \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l) \right]_{j:C_i(j,x_k)>0}, [C_i(j, x_k)]_{j:C_i(j,x_k)>0} \right) \right)^2$$

$$= \sum_{l=1}^{r} \left| \sum_{j:C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \right| \cdot \left| \sum_{j:C_i(j,x_k)>0} C_i(j, x_k)^2 \right| \cdot$$
$$\cdot \left(\cos \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l) \right]_{j:C_i(j,x_k)>0}, [C_i(j, x_k)]_{j:C_i(j,x_k)>0} \right) \right)^2$$

Wszystkie wyrazy pod modułami są $\geqslant 0$

$$= \sum_{l=1}^{r} \sum_{j:C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot \sum_{j:C_i(j,x_k)>0} C_i(j, x_k)^2 \cdot$$
$$\cdot \left(\cos \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l) \right]_{j:C_i(j,x_k)>0}, [C_i(j, x_k)]_{j:C_i(j,x_k)>0} \right) \right)^2$$

Należy zauważyć, że druga równość ma nieco inny charakter od poprzedniego, w szczególności $\sum_{j:C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \leqslant \sum_{j=0}^{N(i)} \tilde{\eta}_{I(i,j)}(x_k)_l^2$, ale zwykle nie są one równe, co pociąga za sobą to, że odpowiednie kąty także zwykle nie są równe:

$$\cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l) \right]_{j:C_i(j,x_k)>0}, [C_i(j, x_k)]_{j:C_i(j,x_k)>0} \right) \geqslant$$
$$\cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l) \right]_{j=0}^{N(i)}, [C_i(j, x_k)]_{j=0}^{N(i)} \right)$$

Dowód twierdzenia o błędach na zbiorze (Tw. 1)

Najpierw skorzystajmy z definicji błędu średniokwadratowego (3.4) oraz lematu 1 (str. 28, dowiedzionego powyżej).

$$\begin{aligned}\mathrm{MSE}(\widetilde{g}_i, \mathbf{S}) = \frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\sum_{j:C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2\cdot C_i(j,x_k)\cdot \\ \cdot\cos^2\left(\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l\cdot\sqrt{C_i(j,x_k)}\right]_{j=0}^{N(i)},\left[\sqrt{C_i(j,x_k)}\right]_{j=0}^{N(i)}\right)\end{aligned}$$

Oznaczając $\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l\cdot\sqrt{C_i(j,x_k)}\right]_{j=0}^{N(i)},\left[\sqrt{C_i(j,x_k)}\right]_{j=0}^{N(i)}$ przez α oraz $\measuredangle\left[\tilde{\eta}_{I(i,j)}(x_k)_l\cdot\sqrt{\overline{C_i(j,x_k)}}\right]_{j=0}^{N(i)},\left[\sqrt{\overline{C_i(j,x_k)}}\right]_{j=0}^{N(i)}$ przez $\overline{\alpha}$, a następnie dodając i odejmując to samo wyrażenie, otrzymujemy:

$$\begin{aligned}=\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\Bigg(\Big(\sum_{C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2\cdot C_i(j,x_k)\Big)\cos^2(\alpha) \\ +\left(\sum_{C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2\frac{1}{n_i^{max}}+\frac{n_i^{max}-n_{ik}}{n_i^{max}}\cdot\eta_i(x_k)_l^2\right)\cos^2(\overline{\alpha}) \\ -\left(\sum_{C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2\frac{1}{n_i^{max}}+\frac{n_i^{max}-n_{ik}}{n_i^{max}}\cdot\eta_i(x_k)_l^2\right)\cos^2(\overline{\alpha})\Bigg)\end{aligned}$$

Po przegrupowaniu:

$$\begin{aligned}=\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\Bigg(\sum_{C_i(j,x_k)>0}\left(\tilde{\eta}_{I(i,j)}(x_k)_l^2\cdot\left(C_i(j,x_k)\cdot\cos^2(\alpha)-\frac{1}{n_i^{max}}\cdot\cos^2(\overline{\alpha})\right)\right) \\ -\frac{n_i^{max}-n_{ik}}{n_i^{max}}\cdot\eta_i(x_k)_l^2\cdot\cos^2(\overline{\alpha}) \\ +\left(\sum_{C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2\frac{1}{n_i^{max}}+\frac{n_i^{max}-n_{ik}}{n_i^{max}}\cdot\eta_i(x_k)_l^2\right)\cdot\cos^2(\overline{\alpha})\Bigg)\end{aligned}$$

Część tego równania jest różnicą błędu pomiędzy daną funkcją, a funkcją opisaną w twierdzeniu, zatem:

$$=\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\left(\tau_{kl}+\left(\sum_{C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2\frac{1}{n_i^{max}}+\frac{n_i^{max}-n_{ik}}{n_i^{max}}\cdot\eta_i(x_k)_l^2\right)\cdot\cos^2(\overline{\alpha})\right)$$

Uwaga, $\sum_{C_i(j,x_k)>0}$ jest warunkiem obejmującym dwie sumy - po k i po j. Korzystając również z jedynki trygonometrzycznej:

$$= \tau + \frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\left(\sum_{C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2\frac{1}{n_i^{max}} + \frac{n_i^{max}-n_{ik}}{n_i^{max}}\cdot\eta_i(x_k)_l^2\right)\cdot\left(1-\sin^2(\overline{\alpha})\right)$$

$$= \tau + \frac{1}{n_i^{max}}\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\left(\sum_{C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2 + (n_i^{max}-n_{ik})\cdot\eta_i(x_k)_l^2\right)$$

$$- \frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\left(\sum_{C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2\frac{1}{n_i^{max}} + \frac{n_i^{max}-n_{ik}}{n_i^{max}}\cdot\eta_i(x_k)_l^2\right)\cdot\sin^2(\overline{\alpha})$$

Ten drugi składnik można określić jako ϱ_{kl}

$$= \tau + \frac{1}{n_i^{max}}\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\left(\sum_{C_i(j,x_k)>0}\tilde{\eta}_{I(i,j)}(x_k)_l^2 + (n_i^{max}-n_{ik})\cdot\eta_i(x_k)_l^2\right)$$

$$- \frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\varrho_{kl}$$

Zmieniając kolejność sumowania:

$$= \tau + \frac{1}{n_i^{max}}\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{C_i(j,x_k)>0}\sum_{l=1}^{r}\tilde{\eta}_{I(i,j)}(x_k)_l^2 + \frac{1}{n_i^{max}}\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}(n_i^{max}-n_{ik})\cdot\eta_i(x_k)_l^2$$

$$- \frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}\varrho_{kl}$$

$$= \tau + \frac{1}{n_i^{max}}\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{C_i(j,x_k)>0}\sum_{l=1}^{r}\tilde{\eta}_{I(i,j)}(x_k)_l^2 + \frac{1}{n_i^{max}}\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}(n_i^{max}-n_{ik})\cdot\eta_i(x_k)_l^2 - \varrho$$

$$= \tau + \frac{1}{n_i^{max}}\sum_{j=0}^{N(i)}\frac{1}{|\mathbf{S}|}\sum_{k:C_i(j,x_k)>0}\sum_{l=1}^{r}\tilde{\eta}_{I(i,j)}(x_k)_l^2 + \frac{1}{n_i^{max}}\frac{1}{|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}(n_i^{max}-n_{ik})\cdot\eta_i(x_k)_l^2 - \varrho$$

$$= \tau + \frac{1}{n_i^{max}}\sum_{j=0}^{N(i)}\frac{|\mathbf{S}_{I(i,j)}|}{|\mathbf{S}|}\frac{1}{|\mathbf{S}_{I(i,j)}|}\sum_{k:C_i(j,x_k)>0}\sum_{l=1}^{r}\tilde{\eta}_{I(i,j)}(x_k)_l^2 +$$

$$+ \frac{1}{n_i^{max}|\mathbf{S}|}\sum_{k=1}^{|\mathbf{S}|}\sum_{l=1}^{r}(n_i^{max}-n_{ik})\cdot\eta_i(x_k)_l^2 - \varrho$$

Z powyższego

$$\mathrm{MSE}(\tilde{g}_i, \mathbf{S}) =$$
$$= \tau + \sum_{j=0}^{N(i)} \frac{|\mathbf{S}_{I(i,j)}|}{n_i^{max} \cdot |\mathbf{S}|} \mathrm{MSE}(\tilde{g}_{I(i,j)}, \mathbf{S}_{I(i,j)}) + \frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} (n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2 - \varrho$$

Przy czym,

$$\begin{aligned}
\tau_{kl} = \sum_{C_i(j,x_k)>0} \Bigg(& \tilde{\eta}_{I(i,j)}(x_k)_l^2 \cdot \\
& \cdot \Bigg(C_i(j,x_k) \cdot \cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)}, \left[\sqrt{C_i(j,x_k)} \right]_{j=0}^{N(i)} \right) + \\
& - \frac{1}{n_i^{max}} \cdot \cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)}, \left[\sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)} \right) \Bigg) \Bigg) \\
& - \frac{n_i^{max} - n_{ik}}{n_i^{max}} \cdot \eta_i(x_k)_l^2 \cdot \\
& \cdot \cos^2 \left(\measuredangle \left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)}, \left[\sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)} \right)
\end{aligned}$$
$$\tau = \frac{1}{|\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \tau_{kl}$$

A także:

$$\begin{aligned}
\varrho_{kl} = & \left(\sum_{C_i(j,x_k)>0} \tilde{\eta}_{I(i,j)}(x_k)_l^2 \frac{1}{n_i^{max}} + \frac{n_i^{max} - n_{ik}}{n_i^{max}} \cdot \eta_i(x_k)_l^2 \right) \cdot \\
& \cdot \sin \left(\left[\tilde{\eta}_{I(i,j)}(x_k)_l \cdot \sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)}, \left[\sqrt{\overline{C_i(j,x_k)}} \right]_{j=0}^{N(i)} \right)^2
\end{aligned}$$
$$\varrho = \frac{1}{|\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \varrho_{kl}$$

□

Dowód twierdzenia z podrozdziału 3.5 (dodawanie poddrzewa)

Należy udowodnić, że przy odpowiednich założeniach (str. 37) błąd węzła po utworzeniu dzieci będzie mniejszy niż błąd aproksymacji w danym węźle.

$$
\begin{aligned}
&\mathrm{MSE}(\widetilde{g}_i, \mathbf{S}) = \\
&\tau + \sum_{j=0}^{N(i)} \frac{|\mathbf{S}_{I(i,j)}|}{n_i^{max} \cdot |\mathbf{S}|} \mathrm{MSE}(\tilde{g}_{I(i,j)}, \mathbf{S}_{I(i,j)}) + \frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} (n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2 - \varrho \\
&\leqslant \tau + \sum_{j=0}^{N(i)} \frac{|\mathbf{S}_{I(i,j)}|}{n_i^{max} \cdot |\mathbf{S}|} \mathrm{MSE}(g_i, \mathbf{S}_{I(i,j)}) + \frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} (n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2 - \varrho \\
&= \tau + \sum_{j=0}^{N(i)} \frac{|\mathbf{S}_{I(i,j)}|}{n_i^{max} \cdot |\mathbf{S}|} \frac{1}{|\mathbf{S}_{I(i,j)}|} \sum_{k:C_i(j,x_k)>0} \sum_{l=1}^{r} \eta_i(x_k)_l^2 + \\
&\qquad \frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} (n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2 - \varrho
\end{aligned}
$$

Z założeń twierdzenia ϱ jest dodatnie, a więc

$$
\begin{aligned}
\mathrm{MSE}(\widetilde{g}_i, \mathbf{S}) < &\sum_{j=0}^{N(i)} \frac{|\mathbf{S}_{I(i,j)}|}{n_i^{max} \cdot |\mathbf{S}|} \frac{1}{|\mathbf{S}_{I(i,j)}|} \sum_{k:C_i(j,x_k)>0} \sum_{l=1}^{r} \eta_i(x_k)_l^2 + \\
&\frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} (n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2
\end{aligned}
$$

zmieniając kolejność sumowania

$$
\begin{aligned}
= &\sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \sum_{j:C_i(j,x_k)>0} \frac{|\mathbf{S}_{I(i,j)}|}{n_i^{max} \cdot |\mathbf{S}|} \frac{1}{|\mathbf{S}_{I(i,j)}|} \eta_i(x_k)_l^2 + \\
&\frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} (n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2
\end{aligned}
$$

łącząc obie sumy i wyciągając stałe

$$
\begin{aligned}
= &\frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \left((n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2 + \sum_{j:C_i(j,x_k)>0} \eta_i(x_k)_l^2 \frac{|\mathbf{S}_{I(i,j)}|}{|\mathbf{S}_{I(i,j)}|} \right) \\
= &\frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \left((n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2 + \eta_i(x_k)_l^2 \sum_{j:C_i(j,x_k)>0} 1 \right) \\
= &\frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \left((n_i^{max} - n_{ik}) \cdot \eta_i(x_k)_l^2 + \eta_i(x_k)_l^2 n_{ik} \right) \\
= &\frac{1}{n_i^{max} \cdot |\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} n_i^{max} \cdot \eta_i(x_k)_l^2 = \frac{1}{|\mathbf{S}|} \sum_{k=1}^{|\mathbf{S}|} \sum_{l=1}^{r} \eta_i(x_k)_l^2 = \mathrm{MSE}(g_i, \mathbf{S})
\end{aligned}
$$

A więc przy założeniach twierdzenia $\mathrm{MSE}(\widetilde{g}_i, \mathbf{S}) < \mathrm{MSE}(g_i, \mathbf{S})$. □

Uzasadnienie przykładu 1 z podrozdziału 3.3

Należy obliczyć, jaka będzie oczekiwana wartość $cos^2(\measuredangle[x_1,\ldots,x_n],[z_1,\ldots,z_n])$ przy ustalonym wektorze x z hipersfery jednostkowej n wymiarowej oraz jednostajnym rozkładzie kierunków z. Kierunki z będą reprezentowane przez wektory y z hipersfery jednostkowej. W takim razie należy policzyć wartość oczekiwaną $cos^2(\measuredangle[x_1,\ldots,x_n],[y_1,\ldots,y_n])$, przy czym y mają jednostajny rozkład na hipersferze jednostkowej.

Na początku należy zauważyć, że wspomniana wartość oczekiwana nie zmienia się przy obrocie całej przestrzeni, a więc nie jest też zależna od konkretnego wyboru wektora x. Wystarczy więc, że zostanie policzona dla jednego konkretnego wektora.

Niech będzie to wektor taki, że wszystkie współrzędne równe są 0 z wyjątkiem ostatniej, równej 1 : $\forall_{i=1,\ldots,n-1} x_i = 0$, $x_n = 1$.

Cosinus wektorów y i x będzie zatem równy ich iloczynowi skalarnemu (oba mają normy równe 1) i będzie to $x_n y_n = y_n$.

Pozostaje pytanie, jaką wagę powinien mieć dany kąt. Ponieważ długość wektora y ma się równać jeden, a zatem też $\sum_{i=1}^{i=n} y_i^2 = 1$ i $\sum_{i=1}^{i=n-1} y_i^2 = 1 - y_n^2$, co będą spełniały wszystkie wektory leżące na sferze będącej brzegiem kuli $n-1$ wymiarowej o promieniu $\sqrt{1-y_n^2}$. Zatem dany kąt powinniśmy uwzględnić proporcjonalnie do powierzchni tej sfery, która zostanie oznaczona jako $S_{n-2}(\sqrt{1-y_n^2})$. Zauważmy, że jeśli y_n jest cosinusem kąta θ, to $\sqrt{1-y_n^2}$ jest modułem z jego sinusa. Ponieważ $\cos^2$ i $|sin|$ są parzyste i mają okres równy π, wystarczy jeśli policzymy wartość oczekiwaną dla dodatnich wartości sin i dodatniej wartości cos (w pozostałych "ćwiartkach" hipersfery będzie taka sama):

$$\frac{\int_{\theta=0}^{\theta=\frac{\pi}{2}} S_{n-2}(\sin\theta)\cos^2\theta d\theta}{\int_{\theta=0}^{\theta=\frac{\pi}{2}} S_{n-2}(\sin\theta) d\theta}$$

Niech C^*_{n-2} oznacza powierzchnię $n-2$ wymiarowej sfery o promieniu 1. Wtedy można zapisać:

$$\frac{\int_{\theta=0}^{\theta=\frac{\pi}{2}} S^*_{n-2}(\sin\theta)^{n-2}\cos^2\theta d\theta}{\int_{\theta=0}^{\theta=\frac{\pi}{2}} S^*_{n-2}(\sin\theta)^{n-2} d\theta}$$

S^*_{n-2} jest niezależne od y_n, więc po prostu można je wyjąć przed całkę i skrócić, a

później przekształcić

$$\frac{\int_{\theta=0}^{\theta=\frac{\pi}{2}}(\sin\theta)^{n-2}\cos^2\theta d\theta}{\int_{\theta=0}^{\theta=\frac{\pi}{2}}(\sin\theta)^{n-2}d\theta}=\frac{\int_{\theta=0}^{\theta=\frac{\pi}{2}}(\sin\theta)^{n-3}\cos^2\sin\theta d\theta}{\int_{\theta=0}^{\theta=\frac{\pi}{2}}(\sin\theta)^{n-3}\sin\theta d\theta}$$

Pamiętając, że w danym zakresie $\sin \geqslant 0$ i $\sin^2 + cos^2 = 1$:

$$\frac{\int_{\theta=0}^{\theta=\frac{\pi}{2}}(1-\cos^2\theta)^{\frac{n-1}{2}-1}\cdot-\frac{1}{2}\cos\theta\cdot-2\cos\theta\sin\theta d\theta}{\int_{\theta=0}^{\theta=\frac{\pi}{2}}(1-\cos^2\theta)^{\frac{n-1}{2}-1}\cdot-\frac{1}{2}\cos^{-1}\theta\cdot-2\cos\theta\sin\theta d\theta}$$

Podstawiając $u = \cos(\theta)^2$ oraz skracając $-\frac{1}{2}$, otrzymuje się:

$$\frac{\int_{u=1}^{u=0}(1-u)^{\frac{n-1}{2}-1}u^{\frac{1}{2}}du}{\int_{u=1}^{u=0}(1-u)^{\frac{n-1}{2}-1}u^{-\frac{1}{2}}du}=\frac{-\int_{u=0}^{u=1}(1-u)^{\frac{n-1}{2}-1}u^{\frac{1}{2}}du}{-\int_{u=0}^{u=1}(1-u)^{\frac{n-1}{2}-1}u^{-\frac{1}{2}}du}=\frac{B(\frac{3}{2},\frac{n-1}{2})}{B(\frac{1}{2},\frac{n-1}{2})}$$

gdzie B oznacza funkcję B Eulera. Przy czym n musi wynosić co najmniej 2. Przekształcając dalej otrzymujemy:

$$\frac{B(\frac{3}{2},\frac{n-1}{2})}{B(\frac{1}{2},\frac{n-1}{2})}=\frac{\Gamma(\frac{3}{2})\Gamma(\frac{n-1}{2})\Gamma(\frac{1}{2}+\frac{n-1}{2})}{\Gamma(\frac{1}{2})\Gamma(\frac{n-1}{2})\Gamma(\frac{3}{2}+\frac{n-1}{2})}=\frac{\frac{1}{2}\Gamma(\frac{1}{2}+\frac{n-1}{2})}{\Gamma(\frac{3}{2}+\frac{n-1}{2})}=\frac{\frac{1}{2}\Gamma(\frac{1}{2}+\frac{n-1}{2})}{\Gamma(\frac{1}{2}+\frac{n-1}{2})(\frac{1}{2}+\frac{n-1}{2})}=\frac{1}{n}$$

co jest wynikiem podanym w przykładzie.

Dowód lematu 3 – przykład z wiedzą o błędach

Na początku przeprowadzone zostanie rozumowanie dla dowolnej liczby węzłów, po czym, gdy rozważania ogólne wymagałyby zbyt skomplikowanych założeń, wcześniej uzyskane fakty zostaną zastosowane do obliczenia końcowego twierdzenia.

Jeśli rozkłady błędów mają łącznie rozkład normalny i ważone są wektorem kompetencji C, to macierz kowariancji wynosi C^TVC a średnie $C \cdot \mu$, gdzie V to macierz kowariancji, a μ to wektor średnich.

W tym miejscu wprowadzone zostaną oznaczenia stosunku kompetencji i-tego węzła do kompetencji pierwszego węzła, który zawsze ma mieć niezerową funkcję kompetencji, zatem w tych oznaczeniach $R_i = \frac{C_i}{C_1}$, a co za tym idzie $R_1 = 1$, $R_i > 0$. Ponadto wektor kompetencji, to wektor R podzielony przez sumę wartości bezwzględnych (metrykę L1) wektora R: $C = \frac{R}{||R||_1}$, czyli $C^TVC = \frac{R^TVR}{||R||_1^2}$. Pochodne tego wektora ze względu na współrzędne można policzyć, korzystając z własności pochodnej (wszystkie potrzebne zachodzą dla tak regularnej funkcji).

$$\frac{\delta}{\delta R_k}\left(\frac{R^T V R}{||R||_1^2}\right) = \frac{\frac{\delta(R^T V R)}{\delta R_k}\cdot ||R||_1^2 - R^T V R \cdot \frac{\delta\left(||R||_1^2\right)}{\delta R_k}}{||R||_1^4} \tag{A.3}$$

$$= \frac{2\left(\sum_i R_i \cdot V_{ki}\right)\left(\sum_i R_i\right)^2 - \sum_i \sum_j R_i R_j V_{ij} \cdot 2\sum_i R_i}{||R||_1^4} \tag{A.4}$$

$$= \frac{2\left(\sum_i R_i \cdot V_{ki}\right)\sum_j R_j - 2\sum_i \sum_j R_i R_j V_{ij}}{||R||_1^3} \tag{A.5}$$

$$= \frac{2\left(\sum_j \sum_i R_j R_i \cdot V_{ki}\right) - 2\sum_i \sum_j R_i R_j V_{ij}}{||R||_1^3} \tag{A.6}$$

$$= \frac{2\left(\sum_i \sum_j R_i R_j \cdot (V_{ik} - V_{ij})\right)}{||R||_1^3} \tag{A.7}$$

Pochodne ze względu na sumę $C \cdot \mu = \frac{R \cdot \mu}{||R||_1}$ wyniosą

$$\frac{\delta}{\delta R_k}\left(\frac{R \cdot \mu}{||R||_1}\right) = \frac{\frac{\delta(R \cdot \mu)}{\delta R_k}||R|| - R \cdot \mu \frac{\delta||R||}{\delta R_k}}{||R||_1^2} \tag{A.8}$$

$$= \frac{\mu_k ||R||_1 - R \cdot \mu}{||R||_1^2} \tag{A.9}$$

$$= \frac{\sum_i R_i(\mu_k - \mu_i)}{||R||_1^2} \tag{A.10}$$

mając ten wzór, można policzyć pochodną względem kwadratu wspomnianej sumy:

$$\begin{aligned}
\frac{\delta}{\delta R_k}\left(\left(\frac{R \cdot \mu}{||R||_1}\right)^2\right) &= \frac{\delta}{\delta R_k}\left(\left(\frac{\sum_i R_i \cdot \mu_i}{\sum_i R_i}\right)^2\right) \\
&= 2\frac{\sum_i R_i \cdot \mu_i}{\sum_i R_i} \cdot \frac{\delta}{\delta R_k}\left(\frac{R \cdot \mu}{||R||_1}\right) \\
&= 2\frac{\sum_i R_i \cdot \mu_i}{\sum_i R_i} \cdot \frac{\sum_i R_i(\mu_k - \mu_i)}{||R||_1^2} \\
&= 2\frac{\sum_j R_j \cdot \mu_j \cdot \sum_i R_i(\mu_k - \mu_i)}{||R||_1^3} \\
&= 2\frac{\sum_i \sum_j R_j R_i \mu_j (\mu_k - \mu_i)}{||R||_1^3}
\end{aligned}$$

Suma kwadratów odchyleń od 0, to suma średniej podniesionej do kwadratu i wariancji $\left(\frac{R \cdot \mu}{||R||}\right)^2 + \frac{R^T V R}{||R||_1^2}$. Przy znajdowaniu optimum funkcji kwadratowej przydatne

jest znalezienie miejsc zerowych pochodnej:

$$\frac{\delta}{\delta R_k}\left(\left(\frac{R\cdot\mu}{||R||}\right)^2+\frac{R^TVR}{||R||_1^2}\right)=\frac{2\left(\sum_i\sum_j R_iR_j\cdot(V_{ik}-V_{ij})\right)}{||R||_1^3}+2\frac{\sum_i\sum_j R_jR_i\mu_j(\mu_k-\mu_i)}{||R||_1^3}$$
$$=2\frac{\left(\sum_i\sum_j R_iR_j\cdot(V_{ik}-V_{ij}+\mu_j(\mu_k-\mu_i))\right)}{||R||_1^3}$$

Wystarczającym warunkiem zerowania pochodnej w tym przypadku jest

$$\left(\sum_i\sum_j R_iR_j\cdot(V_{ik}-V_{ij}+\mu_j(\mu_k-\mu_i))\right)=0$$

Dla liczby węzłów z niezerową kompetencją równej 2, jest tylko jeden ułamek R_2, zatem z zadanymi założeniami:

$$\begin{aligned}(R_2^2(V_{22}-V_{22}+\mu_2(\mu_2-\mu_2))+R_2(V_{12}-V_{12}+\mu_2(\mu_2-\mu_1))\\ +R_2(V_{22}-V_{12}+\mu_1(\mu_2-\mu_2))+(V_{12}-V_{11}+\mu_1(\mu_2-\mu_1)))=0\\ (R_2(\mu_2(\mu_2-\mu_1))+R_2(V_{22}-V_{12})+(V_{12}-V_{11}+\mu_1(\mu_2-\mu_1)))=0\\ (R_2((V_{22}-V_{12})+\mu_2(\mu_2-\mu_1))+(V_{12}-V_{11}+\mu_1(\mu_2-\mu_1)))=0\\ -\frac{V_{12}-V_{11}+\mu_1(\mu_2-\mu_1)}{((V_{22}-V_{12})+\mu_2(\mu_2-\mu_1))}=R_2\\ \frac{V_{11}+\mu_1^2-V_{12}-\mu_1\mu_2}{V_{22}+\mu_2^2-V_{12}-\mu_1\mu_2}=R_2=\frac{C_2}{C_1}\end{aligned}$$

Warto zauważyć, że zgodnie z założeniami i wzorem, przy zmianie R_2 o pewien ε pochodna stanie się niezerowa, ze znakiem zgodnym ze znakiem ε, czyli istotnie w tym punkcie będzie minimum lokalne.

Jak widać, to rozumowanie da się przeprowadzić, z końcowym wynikiem większym od zera, tylko pod pewnymi warunkami, wymienionymi w lemacie.

Wyliczenie błędu, to podstawienie powyższego wyniku (pamiętając, że $R_1=1$) do wzoru wspomnianej wyżej własności: suma kwadratów odchyleń od 0, to suma $\left(\frac{R\cdot\mu}{||R||}\right)^2+\frac{R^TVR}{||R||_1^2}$. W stosunku do swojej wagi zajmuje to wiele miejsca, więc nie zostało tu zapisane.

B. Wyniki testów

W dodatku B prezentowane są wyniki testów na 21 znanych zadaniach. W stosunku do tabel podanych w treści, tu dodano inne miary błędów, m.in. w celu łatwiejszego porównywania wyników z wynikami dostępnymi w literaturze.

Tablica B.1. Wyniki na częściach testujących rozwiązania 1 – proponowanej wersji z ustawieniem domyślnego błędu na 0.

Zadanie	$\overline{RMSE}$	$\sigma RMSE$	$\pm$	$\overline{E_{abs}}$	σE_{abs}	$\overline{NRMSE}$	skalowany
2dplanes	0.998	0.00012	0.00020	0.797	0.00013	0.227	0.082
friedman	1.025	0.00094	0.00163	0.816	0.00077	0.205	0.066
mv	0.085	0.00285	0.00434	0.049	0.00188	0.008	0.004
bank8fm	0.029	0.00002	0.00003	0.021	0.00002	0.188	0.071
elevators	0.002	0.00000	0.00001	0.001	0.00000	0.282	0.057
δelevators	0.001	0.00000	0.00000	0.001	0.00000	0.594	0.109
δailerons	0.00016	0.0000005	0.0000006	0.00011	0.0000005	0.537	0.079
puma8NH	3.16	0.002	0.00337	2.42	0.002	0.561	0.262
kin8nm	0.105	0.00067	0.00119	0.082	0.00050	0.397	0.148
abalone	2.08	0.006	0.00959	1.46	0.003	0.645	0.154
california	55408	81	130	37719	64	0.480	0.228
house8L	30078	67	120	16303	30	0.569	0.120
house16H	32790	103	148	16983	77	0.621	0.131
cpu_act	2.29	0.009	0.015	1.65	0.008	0.125	0.046
cpu_small	2.88	0.007	0.015	2.07	0.005	0.157	0.058
autoMpg	2.54	0.022	0.035	1.81	0.019	0.326	0.139
auto	2700	174	399	1602	58	0.459	0.181
diabetes	0.670	0.02302	0.052	0.516	0.02011	0.929	0.419
boston	2.73	0.072	0.155	1.87	0.033	0.297	0.122
m_cpu	57.4	3.8	7.9	28.8	1.1	0.357	0.100
servo	0.386	0.02554	0.045	0.212	0.01016	0.247	0.112

Tablica B.2. Wyniki na częściach testujących rozwiązania 2 – proponowanej wersji z ustawieniem domyślnego błędu na połowę błędu korzenia.

Zadanie	$\overline{RMSE}$	$\sigma RMSE$	$\pm$	$\overline{E_{abs}}$	σE_{abs}	$\overline{NRMSE}$	skalowany
2dplanes	0.998	0.00009	0.00015	0.797	0.00012	0.227	0.082
friedman	1.026	0.00085	0.00160	0.816	0.00064	0.205	0.066
mv	0.345	0.17590	0.254565	0.241	0.12848	0.033	0.016
bank8fm	0.029	0.00004	0.00007	0.021	0.00003	0.188	0.071
elevators	0.002	0.00001	0.00001	0.001	0.00000	0.282	0.057
δelevators	0.001	0.00000	0.00000	0.001	0.00000	0.594	0.109
δailerons	0.00016	0.0000005	0.0000006	0.00011	0.0000005	0.536	0.079
puma8NH	3.16	0.001	0.00243	2.42	0.001	0.562	0.262
kin8nm	0.105	0.00053	0.00091	0.082	0.00036	0.398	0.148
abalone	2.08	0.006	0.01005	1.46	0.004	0.645	0.154
california	55510	55	114	37777	60	0.481	0.229
house8L	30107	116	196	16317	33	0.570	0.120
house16H	32753	266	638	16943	141	0.620	0.131
cpu_act	2.30	0.007	0.013	1.65	0.004	0.125	0.046
cpu_small	2.88	0.012	0.024	2.06	0.005	0.156	0.058
autoMpg	2.52	0.023	0.038	1.81	0.014	0.323	0.138
auto	2692	255	369	1614	89	0.458	0.180
diabetes	0.659	0.02715	0.051	0.516	0.01819	0.914	0.412
boston	2.75	0.089	0.132	1.87	0.044	0.299	0.122
m_cpu	59.0	5.9	8.2	28.8	1.7	0.367	0.103
servo	0.381	0.02001	0.034	0.217	0.01147	0.244	0.110

Tablica B.3. Wyniki na częściach testujących rozwiązania 3 – proponowanej wersji nie używającej wag przy klastrowaniu.

Zadanie	$\overline{RMSE}$	$\sigma RMSE$	$\pm$	$\overline{E_{abs}}$	σE_{abs}	$\overline{NRMSE}$	skalowany
2dplanes	0.998	0.00008	0.00017	0.797	0.00007	0.227	0.082
friedman	1.026	0.00115	0.00163	0.816	0.00080	0.205	0.066
mv	0.102	0.00183	0.0047712	0.062	0.00140	0.010	0.005
bank8fm	0.029	0.00004	0.00008	0.021	0.00003	0.188	0.071
elevators	0.002	0.00000	0.00001	0.001	0.00000	0.281	0.057
δelevators	0.001	0.00000	0.00000	0.001	0.00000	0.595	0.109
δailerons	0.00016	0.0000005	0.0000007	0.00011	0.0000003	0.537	0.079
puma8NH	3.16	0.002	0.00366	2.42	0.001	0.562	0.262
kin8nm	0.105	0.00056	0.00119	0.082	0.00046	0.399	0.149
abalone	2.07	0.005	0.01080	1.46	0.003	0.644	0.154
california	55488	56	125	37781	61	0.481	0.229
house8L	30683	119	269	16793	69	0.581	0.123
house16H	33150	167	257	17376	102	0.627	0.133
cpu_act	2.30	0.011	0.017	1.65	0.004	0.125	0.047
cpu_small	2.89	0.007	0.013	2.08	0.004	0.157	0.058
autoMpg	2.53	0.025	0.041	1.81	0.018	0.324	0.138
auto	2756	285	440	1628	94	0.469	0.184
diabetes	0.667	0.02295	0.035	0.514	0.01737	0.926	0.417
boston	2.71	0.059	0.117	1.89	0.027	0.295	0.121
m_cpu	60.5	6.8	11.3	30.0	1.8	0.376	0.106
servo	0.381	0.01834	0.037	0.220	0.01082	0.244	0.111

Tablica B.4. Wyniki na częściach testujących porównywanego rozwiązania 1 – wybrana sieć neuronowa.

Zadanie	$\overline{RMSE}$	$\sigma RMSE$	$\pm$	$\overline{E_{abs}}$	σE_{abs}	$\overline{NRMSE}$	skalowany
2dplanes	0.999	0.00017	0.00035	0.798	0.00022	0.227	0.082
friedman	1.008	0.00068	0.00126	0.802	0.00065	0.202	0.065
mv	0.245	0.01109	0.019185	0.167	0.00637	0.024	0.011
bank8fm	0.028	0.00013	0.00020	0.021	0.00010	0.187	0.071
elevators	0.002	0.00003	0.00006	0.001	0.00000	0.285	0.058
δelevators	0.001	0.00000	0.00000	0.001	0.00000	0.597	0.109
δailerons	0.00016	0.0000009	0.0000019	0.00011	0.0000004	0.538	0.079
puma8NH	3.16	0.003	0.00587	2.42	0.003	0.561	0.262
kin8nm	0.073	0.00041	0.00082	0.056	0.00025	0.275	0.103
abalone	2.07	0.009	0.01852	1.46	0.005	0.642	0.153
california	52986	107	187	35834	70	0.459	0.218
house8L	29887	190	335	16349	78	0.566	0.120
house16H	33352	184	372	18120	102	0.631	0.133
cpu_act	2.45	0.042	0.088	1.71	0.007	0.133	0.050
cpu_small	2.96	0.050	0.090	2.07	0.011	0.161	0.060
autoMpg	2.89	0.066	0.108	2.05	0.034	0.370	0.158
auto	3234	347	648	1978	149	0.550	0.216
diabetes	0.663	0.03110	0.066	0.508	0.02531	0.920	0.414
boston	3.55	0.151	0.266	2.44	0.080	0.387	0.158
m_cpu	79.2	7.8	12.4	36.1	1.9	0.493	0.139
servo	0.471	0.03908	0.061	0.293	0.01655	0.302	0.137

Tablica B.5. Wyniki na częściach testujących rozwiązania porównywanego 2 – prostego komitetu.

Zadanie	$\overline{RMSE}$	$\sigma RMSE$	$\pm$	$\overline{E_{abs}}$	σE_{abs}	$\overline{NRMSE}$	skalowany
2dplanes	0.998	0.00003	0.00007	0.797	0.00003	0.227	0.082
friedman	1.045	0.00105	0.00221	0.831	0.00079	0.209	0.067
mv	0.184	0.00112	0.002	0.115	0.001	0.018	0.008
bank8fm	0.029	0.00001	0.00002	0.021	0.00001	0.190	0.072
elevators	0.002	0.00001	0.00001	0.001	0.00000	0.289	0.059
δelevators	0.001	0.00000	0.00000	0.001	0.00000	0.596	0.109
δailerons	0.00016	0.0000000	0	0.00012	0.0000000	0.542	0.080
puma8NH	3.15	0.001	0.00093	2.42	0.001	0.561	0.262
kin8nm	0.124	0.00014	0.00021	0.097	0.00011	0.470	0.175
abalone	2.05	0.001	0.00208	1.44	0.001	0.635	0.152
california	57411	30	77	39698	24	0.498	0.237
house8L	31260	36	75	17237	25	0.592	0.125
house16H	33103	309	874	17228	97	0.626	0.132
cpu_act	2.32	0.037	0.104	1.68	0.018	0.126	0.047
cpu_small	2.95	0.028	0.079	2.13	0.012	0.160	0.060
autoMpg	2.58	0.141	0.400	1.85	0.087	0.330	0.141
auto	2317	529	1492	1430	282	0.394	0.155
diabetes	0.630	0.07300	0.207	0.491	0.05533	0.875	0.394
boston	2.85	0.308	0.875	1.95	0.165	0.310	0.127
m_cpu	52.6	10.2	29.0	27.6	3.9	0.327	0.092
servo	0.398	0.04741	0.134	0.239	0.02089	0.255	0.115

Tablica B.6. Wyniki na częściach testujących rozwiązania porównywanego 3 – *Bootstrap aggregating*.

Zadanie	$\overline{RMSE}$	$\sigma RMSE$	$\pm$	$\overline{E_{abs}}$	σE_{abs}	$\overline{NRMSE}$	skalowany
2dplanes	0.999	0.00013	0.00021	0.798	0.00012	0.227	0.082
friedman	1.120	0.03097	0.07743	0.887	0.02227	0.224	0.072
mv	0.221	0.00548	0.01124	0.144	0.00383	0.021	0.010
bank8fm	0.029	0.00002	0.00004	0.021	0.00001	0.190	0.072
elevators	0.002	0.00002	0.00003	0.001	0.00000	0.299	0.061
δelevators	0.001	0.00000	0.00000	0.001	0.00000	0.595	0.109
δailerons	0.00016	0.0000003	0.0000009	0.00012	0.0000003	0.539	0.080
puma8NH	3.15	0.001	0.00156	2.42	0.001	0.561	0.262
kin8nm	0.117	0.00051	0.00122	0.091	0.00034	0.443	0.165
abalone	2.05	0.003	0.00453	1.45	0.002	0.636	0.152
california	57007	95	163	39342	75	0.494	0.235
house8L	30991	43	82	17025	41	0.586	0.124
house16H	34705	147	249	18121	91	0.657	0.139
cpu_act	2.41	0.015	0.025	1.75	0.004	0.131	0.049
cpu_small	2.99	0.009	0.020	2.17	0.003	0.162	0.060
autoMpg	2.60	0.015	0.029	1.84	0.019	0.333	0.142
auto	2623	99	179	1550	53	0.446	0.175
diabetes	0.676	0.01278	0.021	0.559	0.01579	0.938	0.423
boston	3.16	0.052	0.095	2.12	0.024	0.344	0.141
m_cpu	59.1	1.8	2.8	28.7	0.6	0.368	0.103
servo	0.520	0.011	0.0132	0.267	0.0066	0.333	0.151

Tablica B.7. Wyniki na częściach testujących rozwiązania porównywanego 4 – wzmacniania gradientowego.

Zadanie	$\overline{RMSE}$	$\sigma RMSE$	$\pm$	$\overline{E_{abs}}$	σE_{abs}	$\overline{NRMSE}$	skalowany
2dplanes	1.000	0.00017	0.00035	0.798	0.00014	0.228	0.082
friedman	1.004	0.00052	0.00095	0.799	0.00038	0.201	0.065
mv	0.156	0.00047	0.00083	0.092	0.00092	0.015	0.007
bank8fm	0.028	0.00004	0.00008	0.021	0.00004	0.186	0.070
elevators	0.002	0.00021	0.00039	0.001	0.00001	0.313	0.064
δelevators	0.001	0.00000	0.00000	0.001	0.00000	0.594	0.108
δailerons	0.00016	0.0000004	0.0000008	0.00011	0.0000005	0.532	0.079
puma8NH	3.15	0.002	0.00344	2.44	0.001	0.561	0.262
kin8nm	0.070	0.00034	0.00052	0.054	0.00024	0.267	0.099
abalone	2.06	0.005	0.00898	1.46	0.003	0.640	0.153
california	51673	167	378	34811	109	0.448	0.213
house8L	29327	75	129	15841	40	0.555	0.117
house16H	32964	97	167	17026	58	0.624	0.132
cpu_act	2.28	0.019	0.038	1.62	0.003	0.124	0.046
cpu_small	2.77	0.013	0.023	1.97	0.006	0.150	0.056
autoMpg	2.57	0.047	0.090	1.79	0.030	0.329	0.140
auto	2833	320	539	1605	122	0.482	0.189
diabetes	0.801	0.03696	0.082	0.634	0.02228	1.112	0.501
boston	2.81	0.063	0.114	1.91	0.031	0.306	0.125
m_cpu	52.5	5.3	9.5	28.9	1.5	0.327	0.092
servo	0.428	0.001	0.0136	0.234	0.001	0.247	0.124

Tablica B.8. Zużycie czasu procesora (suma zużycia czasu rdzeni) AMD Opteron 8350 (2GHz) dla całej procedury testowej (10x10-krotna walidacja), dla metod łączenia rozwiązań i "dużych" zbiorów. Uzyskane poleceniem **time**.

Rozwiązanie					
	Proponowane		Istniejące		
Zadanie	1 ("błąd 0")	2("$\frac{1}{2}$ korzenia")	komitet	bagging	(Wzm. Grad.)[1]
2dplanes	117744	121951	104479	85008	84248
friedman	157136	159506	105803	87903	84071
mv	160498	25767	127787	86104	48288
bank8fm	14435	14708	13621	11825	12112
elevators	122393	122846	106630	62531	68582
δelevators	14330	14551	14559	10183	10346
δailerons	7720	8457	9218	6663	6760
puma8NH	14176	14799	13637	12561	11762
kin8nm	15061	15266	13290	11331	11838
abalone	7328	7582	8621	6534	6828
california	35183	35933	33401	32547	30503
house8L	42375	44542	35709	35482	33940
house16H	135315	142809	79419	71477	73306
cpu_act	83925	26724	67117	35710	39939
cpu_small	24839	27502	20932	15743	17953
Suma	952458	782942	754222	571603	540477

[1]Szukanie wartości parametru wzmacniania gradientowego nie zostało wliczone. Dokładniej: liczba w tabeli powstała poprzez podzielenie przez 2.5 czasu całej procedury, łącznie z testowaniem wartości parametru.

Tablica B.9. $\overline{NMRSE}$ na zbiorach treningowych.

Rozwiązanie	Proponowane			Istniejące			
Zadanie	1	2	3	1(SSN)	2(Kom.)	3(Bag.)	4(WG)
2dplanes	0.226	0.226	0.226	0.227	0.227	0.227	0.225
friedman	0.204	0.204	0.204	0.200	0.209	0.224	0.199
mv	0.008	0.033	0.010	0.023	0.226	0.039	0.047
bank8fm	0.181	0.182	0.183	0.178	0.188	0.187	0.174
elevators	0.269	0.270	0.271	0.268	0.283	0.294	0.260
δelevators	0.584	0.584	0.585	0.587	0.592	0.590	0.582
δailerons	0.523	0.522	0.526	0.512	0.534	0.532	0.490
puma8NH	0.549	0.549	0.548	0.556	0.558	0.557	0.546
kin8nm	0.387	0.388	0.389	0.265	0.467	0.439	0.240
abalone	0.584	0.584	0.585	0.625	0.616	0.618	0.593
california	0.474	0.475	0.475	0.445	0.496	0.492	0.423
house8L	0.547	0.547	0.569	0.534	0.586	0.580	0.523
house16H	0.580	0.577	0.601	0.570	0.610	0.644	0.583
cpu_act	0.115	0.114	0.115	0.116	0.121	0.126	0.108
cpu_small	0.149	0.149	0.150	0.137	0.156	0.159	0.134
autoMpg	0.225	0.226	0.229	0.262	0.279	0.272	0.217
auto	0.141	0.142	0.133	0.227	0.140	0.172	0.067
diabetes	0.570	0.572	0.572	0.659	0.587	0.504	0.465
boston	0.160	0.160	0.160	0.247	0.214	0.244	0.154
m cpu	0.135	0.138	0.140	0.128	0.146	0.201	0.126
servo	0.127	0.129	0.132	0.195	0.164	0.197	0.123
Średnia	0.321	0.322	0.324	0.331	0.342	0.346	0.298

Najbardziej przydatne oznaczenia

x – wektor wejściowy, x_k – k-ty wektor wejściowy w danym zbiorze przykładów.

y – oczekiwane wyjście, y_k – k-te oczekiwane wyjście w danym zbiorze przykładów.

$\mathbf{S}, \mathbf{T}, \mathbf{U}$ – odpowiednio: zbiór przykładów, zbiór testowy, zbiór uczący.

$|\mathbf{S}|$ – liczność zbioru przykładów.

MSE – błąd średniokwadratowy.

NRMSE – znormalizowany (podzielony przez odchylenie standardowe wartości funkcji) pierwiastek błędu średniokwadratowego.

i – zwykle indeks węzła, którego dotyczy dany wzór itp.

$N(i)$ liczba bezpośrednich potomków węzła o indeksie i.

$C_i(j, x_k)$ wartość funkcji kompetencji w węźle i dla j-tego bezpośredniego potomka i wektora wejściowego x_k

$C_i(0, x_k)$ wartość funkcji kompetencji dla aproksymacji w węźle i.

$I(i, j)$ indeks j-tego bezpośredniego potomka węzła i.

g_i – aproksymacja funkcji rozwiązaniem składowym w węźle i.

$\widetilde{g}_i$ – aproksymacja funkcji przez całe poddrzewo węzła i.

$\widetilde{g}_{I(i,j)}$ – aproksymacja funkcji przez poddrzewo j-tego potomka węzła i.

$\eta_i(x_k) = g_i(x_k) - y_k$ – różnica pomiędzy odpowiedzią aproksymacji w węźle i a prawdziwą wartością wyjścia.

$\tilde{\eta}_i(x_k) = \widetilde{g}_i(x_k) - y_k$ – różnica pomiędzy odpowiedzią poddrzewa aproksymacji zakorzenionego w węźle i a prawdziwą wartością wyjścia.

$\measuredangle A, B$ – kąt pomiędzy wektorami A i B.

$[a_i]_{i=0}^n$ – wektor o współrzędnych a_i dla $i = 0, \ldots n$.

Bibliografia

[1] Repozytorium StatLib. http://lib.stat.cmu.edu.

[2] Strona danych metereologicznych Interdyscyplinarnego Centrum Modelowania Uniwersytetu Warszawskiego. http://www.meteo.pl/.

[3] Strona danych metereologicznych Interdyscyplinarnego Centrum Modelowania Uniwersytetu Warszawskiego (często zadawane pytania). http://www.meteo.pl/faq/index.php.

[4] Strona internetowa Towarowej Giełdy Energii. http://tge.pl.

[5] Strona internetowa Urzędu Regulacji Energetyki. http://www.ure.gov.pl.

[6] Strona Polskich Sieci Elektroenergetycznych. http://www.pse.pl/.

[7] Strona Tesla Forecast. http://www.teslaforecast.com/TeslaModel.aspx.

[8] Strona zbiorów danych Waikato Environment for Knowledge Analysis. http://www.cs.waikato.ac.nz/ml/weka/datasets.html.

[9] Instrukcja ruchu i eksploatacji sieci przesyłowej, 2014.

[10] Szczegółowe zasady obrotu i rozliczeń dla energii elektrycznej na rynku dnia bieżącego, 2014.

[11] AlRashidi, M., EL-Naggar, K. Long term electric load forecasting based on particle swarm optimization. *Applied Energy 87*, 1 (2010), s. 320 - 326.

[12] Avnimelech, R., Intrator, N. Boosting regression estimators. *Neural Computation 11*, 2 (1999), s. 499–520.

[13] Bahrami, S., Hooshmand, R.-A., Parastegari, M. Short term electric load forecasting by wavelet transform and grey model improved by {PSO} (particle swarm optimization) algorithm. *Energy 72* (2014), s. 434 - 442.

[14] Bąk, M., Bielecki, A. Neural systems for short-term forecasting of electric power load. W *Adaptive and Natural Computing Algorithms* (2007)Edytorzy: , B. Beliczynski, A. Dzielinski, M. Iwanowski, B. Ribeiro *LNCS*, tom 4432, Springer-Verlag, s. 133–142.

[15] Bezdek, J. C., Keller, J. M., Krishnapuram, R., Pal, N. R. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.* Springer, 1999.

[16] Breiman, L. Bagging predictors. *MachineLearning 24(3)* (1996).

[17] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), s. 5-32.

[18] Breiman, L. Using iterated bagging to debias regressions. *Machine Learning 45*, 3 (2001), s. 261-277.

[19] Breiman, L., Friedman, J., Olshen, R. A., Stone, C. J. *Classification and Regression Trees.* Chapman and Hall/CRC, 1984.

[20] Brodowski, S. Adaptujący się hierarchiczny aproksymator. Msc, Jagiellonian University, 2007. MSc thesis (in Polish).

[21] Brodowski, S. On mean squared error of hierarchical estimator. *Schedae Informaticae 20* (2011), s. 83–99.

[22] Brodowski, S. A validity criterion for fuzzy clustering. W *Computational Collective Intelligence, Technologies and Applications, Proceedings of third ICCCI* (2011)

Lecture Notes in Artificial Intelligence, tom 6922, Springer, s. 113–122.

[23] Brodowski, S., Bielecki, A. New specifics for a hierarchial estimator meta-algorithm. W *Proceedings of the 11th International Conference on Artificial Intelligence and Soft Computing - Volume Part II* (Berlin, Heidelberg, 2012) *Lecture Notes in Computer Science*, tom 7268, Springer-Verlag, s. 22–29.

[24] Brodowski, S., Podolak, I. T. Hierarchical estimator. *Expert Systems with Applications 38*, 10 (2011), s. 12237–12248.

[25] Brown, G., Wyatt, J. L., Tiňo, P. Managing diversity in regression ensembles. *J. Mach. Learn. Res. 6* (2005), s. 1621–1650.

[26] Camacho, R. Laboratory note. learning to turn: decision-trees to perform a levelled turn. Tech. rep., Oxford University Computing Laboratory, 1995.

[27] Camacho, R. Laboratory note. learning to turn: Parameterised decision trees to perform a levelled turn. Tech. rep., 1997.

[28] Chou, J.-S., Tsai, C.-F. Preliminary cost estimates for thin-film transistor liquid-crystal display inspection and repair equipment: A hybrid hierarchical approach. *Comput. Ind. Eng. 62*, 2 (2012), s. 661–669.

[29] Chou, J.-S., Tsai, C.-F., Pham, A.-D., Lu, Y.-H. Machine learning in concrete strength simulations: Multi-nation data analytics. *Construction and Building Materials 73* (2014), s. 771 - 780.

[30] Clemen, R. T. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting 5*, 4 (1989), s. 559 - 583.

[31] de Souza, F. J., Vellasco, M. M. R., Pacheco, M. A. C. Hierarchical neuro-fuzzy quadtree models. *Fuzzy Sets Syst. 130*, 2 (2002), s. 189–205.

[32] Drucker, H. Improving regressors using boosting techniques. W *Proceedings of the Fourteenth International Conference on Machine Learning* (San Francisco, CA, USA, 1997), ICML '97, Morgan Kaufmann Publishers Inc., s. 107–115.

[33] Duchi, J., Singer, Y. Boosting with structural sparsity. W *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ACM, s. 297–304.

[34] Duffy, N., Helmbold, D. Boosting methods for regression. *Machine Learning 47*, 2-3 (2002), s. 153-200.

[35] Feinberg, E., Genethliou, D. Load forecasting. W *Applied Mathematics for Restructured Electric Power Systems*, Edytorzy: J. Chow, F. Wu, J. Momoh, Power Electronics and Power Systems. Springer US, 2005, s. 269–285.

[36] Frayman, Y., Rolfe, B. F., Webb, G. I. Solving regression problems using competitive ensemble models. W *AI 2002: Advances in Artificial Intelligence*. Springer Berlin Heidelberg, 2002, s. 511–522.

[37] Freund, Y., Schapire, R. A decision theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences 55* (1997), s. 119–139.

[38] Friedman, J. Multivariate adaptative regression splines. *Annals of Statistics 19:1* (1991), s. 1–141.

[39] Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29* (2000), s. 1189–1232.

[40] Friedman, J. H. Stochastic gradient boosting. *Comput. Stat. Data Anal. 38*, 4 (2002), s. 367–378.

[41] Ghahramani, Z. The kin datasets, 1996. http://www.cs.toronto.edu/~delve/data/kin/kin.ps.gz.

[42] Granitto, P., Verdes, P., Ceccatto, H. Neural network ensembles: evaluation

of aggregation algorithms. *Artificial Intelligence 163*, 2 (2005), s. 139 - 162.

[43] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. The weka data mining software: An update. *SIGKDD Explor. Newsl. 11*, 1 (2009), s. 10–18.

[44] Hand, D., Mannila, H., Smyth, P. *Principles of Data Mining*. MIT Press, 2001.

[45] Harrison, D., Rubinfeld, D. L. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management 5*, 1 (1978), s. 81-102.

[46] Hashem, S. Optimal linear combinations of neural networks. *Neural Networks 10*, 4 (1997), s. 599 - 614.

[47] Hastie, T., Tibshirani, R. *Generalized Additive Models*. John Wiley & Sons, Ltd, 2014.

[48] Hastie, T., Tibshirani, R., Friedman, J. *The Elements of Statistical Learning*, 2 ed. Springer-Verlag New York, 2009.

[49] Hu, Z., Bao, Y., Xiong, T. Comprehensive learning particle swarm optimization based memetic algorithm for model selection in short-term load forecasting using support vector regression. *Applied Soft Computing 25*, 0 (2014), s. 15 - 25.

[50] Huang, G., Huang, G., Song, S., You, K. Trends in extreme learning machines: A review. *Neural Networks 61* (2015), s. 32–48.

[51] Islam, M. M., Yao, X., Murase, K. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks 14* (2003), s. 820–834.

[52] Jiang, W., Tanner, M. A. Hierarchical mixtures-of-experts for exponential family regression models: Approximation and maximum likelihood estimation. *Ann. Statistics 27* (1999), s. 987–1011.

[53] Jordan, M. I., Jacobs, R. A. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* (1994), s. 181–214.

[54] Kelley Pace, R., Barry, R. Sparse spatial autoregressions. *Statistics & probability letters 33*, 3 (1997), s. 291–297.

[55] Khwaja, A., Naeem, M., Anpalagan, A., Venetsanopoulos, A., Venkatesh, B. Improved short-term load forecasting using bagged neural networks. *Electric Power Systems Research 125*, 0 (2015), s. 109 - 115.

[56] Kruse, R., Döring, C., Lesot, M.-J. *Fundamentals of Fuzzy Clustering*. John Wiley & Sons, Ltd, 2007, s. 1–30.

[57] Kulkarni, S., Simon, S. P., Sundareswaran, K. A spiking neural network (SNN) forecast engine for short-term electrical load forecasting. *Applied Soft Computing 13*, 8 (2013), s. 3628 - 3635.

[58] Kuncheva, L., Rodríguez, J. An experimental study on rotation forest ensembles. W *Multiple Classifier Systems*, Edytorzy: M. Haindl, J. Kittler, F. Roli *Lecture Notes in Computer Science*, tom 4472. Springer Berlin Heidelberg, 2007, s. 459–468.

[59] Kyriakides, E., Polycarpou, M. Short term electric load forecasting: A tutorial. W *Trends in Neural Computation Studies in Computational Intelligence*, tom 35. Springer Berlin / Heidelberg, 2007, s. 391–418.

[60] LeCun, Y., Bottou, L., Orr, G., Müller, K.-R. Efficient backprop. W *Neural Networks: Tricks of the Trade*, Edytorzy: G. Orr K.-R. Müller *Lecture Notes in Computer Science*, tom 1524. Springer Berlin Heidelberg, 1998, s. 9–50.

[61] Lendasse, A., Cottrell, M., Wertz, V., Verleysen, M. Prediction of electric load using kohonen maps - application to the polish electricity consumption. W *Proceedings of the American Control Conference* (2002), s. 3684–3689.

[62] Liaw, A., Wiener, M. Classification and regression by randomforest. *R News 2*, 3 (2002), s. 18-22.
[63] Lichman, M. UCI machine learning repository, 2013. http://archive.ics.uci.edu/ml.
[64] Liu, Y., Yao, X. Ensemble learning via negative correlation. *Neural Networks 12*, 10 (1999), s. 1399 - 1404.
[65] Majchrzak, H., Mroziński, A., Pozniak, R. Wpływ funkcjonowania rynku bilansującego na koszty ponoszone przez uczestników rynku energii elektrycznej. *Energetyka* (2005).
[66] Meir, R., Rätsch, G. An introduction to boosting and leveraging. W *Advanced Lectures on Machine Learning*, Edytorzy: S. Mendelson A. Smola *Lecture Notes in Computer Science*, tom 2600. Springer Berlin Heidelberg, 2003, s. 118–183.
[67] Mendes-Moreira, J. a., Soares, C., Jorge, A. M., Sousa, J. F. D. Ensemble approaches for regression: A survey. *ACM Comput. Surv. 45*, 1 (2012), s. 10:1–10:40.
[68] Mooij, J., Janzing, D. Distinguishing between cause and effect. *Journal of Machine Learning Research Workshop and Conference Proceedings 6* (2010), s. 147–156.
[69] Musiałkiewicz, l., Grzejszczak, P., Skoczek, S., Kosiarski, K., Michalczyk, P., Michalak, K. Raport o rynku energii elektrycznej i gazu ziemnego w polsce w 2014 roku, 2015.
[70] Nash, W. *The Population Biology of Abalone (Haliotis Species) in Tasmania: Blacklip abalone (H. rubra) from the north coast and the islands of Bass Strait.* Nr. v. 1 w Technical report (Tasmania. Sea Fisheries Division). Sea Fisheries Division, Marine Research Laboratories - Taroona, Department of Primary Industry and Fisheries, Tasmania, 1994.
[71] Nie, H., Liu, G., Liu, X., Wang, Y. Hybrid of {ARIMA} and {SVMs} for short-term load forecasting. *Energy Procedia 16, Part C* (2012), s. 1455 - 1460. 2012 International Conference on Future Energy, Environment, and Materials.
[72] Pal, N., Bezdek, J. On cluster validity for the fuzzy c-means model. *IEEE Transactions on Fuzzy Systems 3*, 3 (1995), s. 370–379.
[73] Pardo, C., Diez-Pastor, J. F., García-Osorio, C., Rodríguez, J. J. Rotation forests for regression. *Applied Mathematics and Computation 219*, 19 (2013), s. 9914 - 9924.
[74] Pedrycz, W., Rai, P. Experience-consistent modeling: Regression and classification problems. *Automatica 45*, 2 (2009), s. 449 - 455.
[75] Podolak, I. T. Hierarchical rules for a hierarchical classifier. W *Adaptive and Natural Computing Algorithms* (2007)Edytorzy: , B. Beliczynski, A. Dzielinski, M. Iwanowski, B. Ribeiro *LNCS*, tom 4431, Springer-Verlag, s. 749–757.
[76] Podolak, I. T. Hierarchical classifier with overlapping class groups. *Expert Systems with Applications 34*, 1 (2008), s. 673–682.
[77] Podolak, I. T., Roman, A. A new notion of weakness in classification theory. W *Computer Recognition Systems*, Edytorzy: Kurzynski W. et. al. *Advances in Soft Computing*, tom 57. Springer Berlin / Heidelberg, 2009, s. 239–245.
[78] Poland, J. On the robustness of update strategies for the bayesian hyperparameter α, 2001.
[79] Pousinho, H., Mendes, V., Catalão, J. Short-term electricity prices forecasting in a competitive market by a hybrid PSO–ANFIS approach. *International Journal of Electrical Power & Energy Systems 39*, 1 (2012), s. 29 - 35.
[80] Quinlan, J. R. Learning with continuous classes. W *Proceedings of the 5-th Austra-*

lian Conference on Artificial Intelligence, AI'92, World Scientific (1992), s. 343–348.

[81] Quinlan, J. R. Combining instance-based and model-based learning. W *Proceedings of the Tenth International Conference on Machine Learning* (1993), Morgan Kaufmann, s. 236–243.

[82] Ramamurti, V., Ghosh, J. Structurally adaptive localized mixtures of experts for non-stationary environments.

[83] Rasmussen, C. E., Neal, R. M., Hinton, G., van Camp, D., Revow, M., Ghahramani, Z., Kustra, R., Tibshirani, R. Data for evaluating learning in valid experiments, 2002. http://www.cs.toronto.edu/∼delve/.

[84] Reiner, P., Wilamowski, B. M. Efficient incremental construction of {RBF} networks using quasi-gradient method. *Neurocomputing 150, Part B*, 0 (2015), s. 349 - 356. Special Issue on Information Processing and Machine Learning for Applications of Engineering.

[85] Ridgeway, G., Madigan, D., Richardson, T. Boosting methodology for regression problems. W *The Seventh International Workshop on Artificial Intelligence and Statistics (Uncertainty '99)* (1999), Morgan Kaufmann, s. 152–161.

[86] Rokach, L. Ensemble-based classifiers. *Artificial Intelligence Review 33*, 1-2 (2010), s. 1-39.

[87] Russell, S. J., Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[88] Saito, K., Nakano, R. A constructive learning algorithm for an HME. W *IEEE International Conference on Neural Networks, 1996* (1996), tom 3, s. 1268–1273.

[89] Schapire, R. E. The strength of weak learnability. *Machine Learning 5*, 2 (1990), s. 197–227.

[90] Shrestha, D. L., Solomatine, D. P. Experiments with AdaBoost.RT, an improved boosting scheme for regression. *Neural Computation 18*, 7 (2006), s. 1678-1710.

[91] Sudheer, G., Suseelatha, A. Short term load forecasting using wavelet transform combined with Holt–Winters and weighted nearest neighbor models. *International Journal of Electrical Power & Energy Systems 64* (2015), s. 340 - 346.

[92] Thomopoulos, S., Bougoulias, D., Wann, C.-D. On cluster validity for the fuzzy c-means model. *IEEE Transactions on Aerospace and Electronic Systems 31*, 1 (1995), s. 21 – 38.

[93] Tresp, V. Committee machines. W *Handbook for Neural Network Signal Processing*, Edytorzy: Y. H. Hu J.-N. Hwang. CRC Press, 2001.

[94] Vapnik, V. *The Nature of Statistical Learning Theory*. Springer, 1995.

[95] Wang, D., Zeng, X., Keane, J. A. Hierarchical hybrid fuzzy-neural networks for approximation with mixed input variables. *Neurocomputing 70*, 16-18 (2007), s. 3019–3033.

[96] Waterhouse, S., Robinson, A. Constructive algorithms for hierarchical mixtures of experts, 1995.

[97] Wilamowski, B., Yu, H. Improved computation for Levenberg–Marquardt training. *Neural Networks, IEEE Transactions on 21*, 6 (2010), s. 930-937.

[98] Winkler, R., Klawonn, F., Kruse, R. Fuzzy c-means in high dimensional spaces. *International Journal of Fuzzy System Applications 1*, 1 (2011), s. 1–16.

[99] Wolberg, W. H., Streetand, W. N., Mangasarian., O. L. Computerized diagnosis of breast fine-needle aspirates. *The Breast Journal 3*, 2 (1997), s. 77-80.

[100] Wolpert, D. H. The lack of a priori distinctions between learning algorithms. *Neural Computation 8*, 7 (1996), s. 1341–1390.

[101] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H., Steinbach, M., Hand, D., Steinberg, D. Top 10 algorithms in data mining. *Knowledge and Information Systems 14*, 1 (2008), s. 1-37.

[102] Yao, J., Tan, C. L. A case study on using neural networks to perform technical forecasting of forex. *Neurocomputing 34* (2000), s. 79–98.

[103] Yu, L., Wang, S., Lai, K. K. A neural-network-based nonlinear metamodeling approach to financial time series forecasting. *Applied Soft Computing 9*, 2 (2009), s. 563 - 574.

[104] Zhou, Z.-H., Wu, J., Tang, W. Ensembling neural networks: Many could be better than all. *Artificial Intelligence 137*, 1–2 (2002), s. 239 - 263.