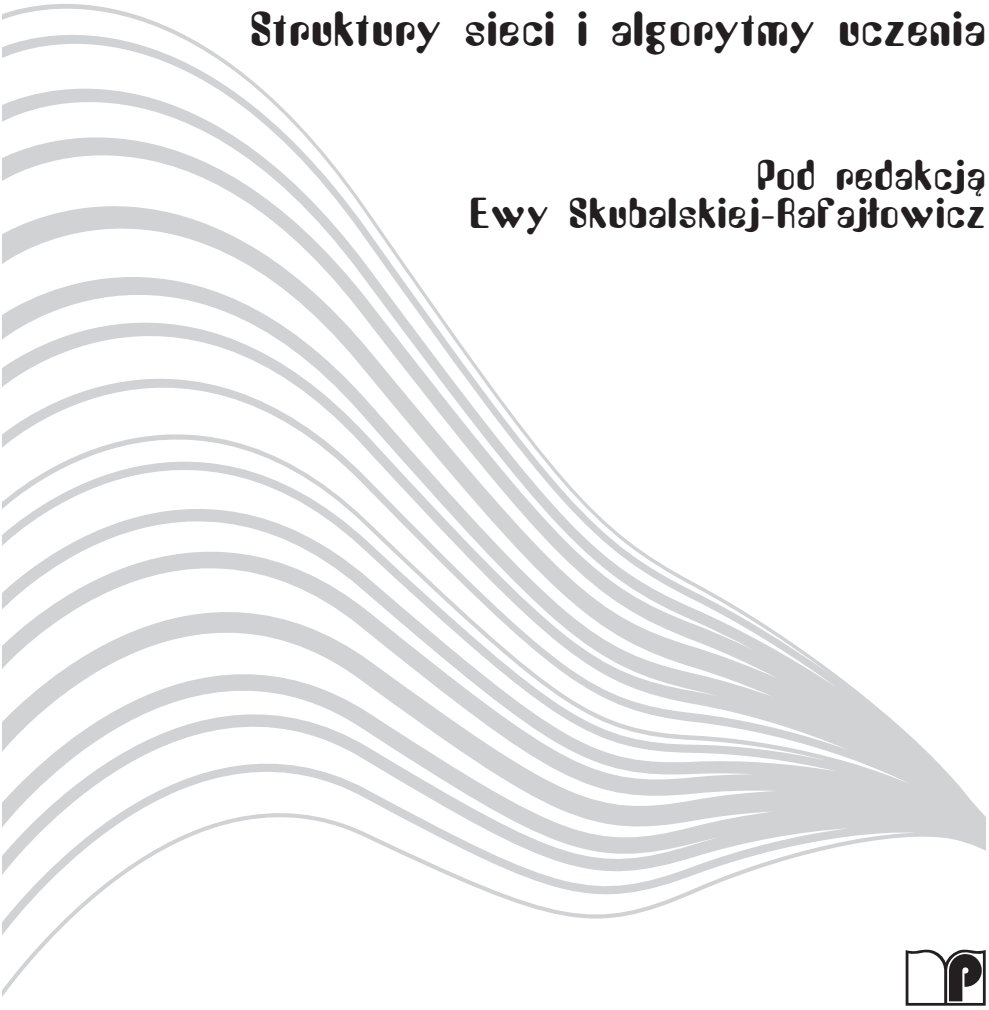
Sieci neuronowe w przetwarzaniu strumieni danych

Struktury sieci i algorytmy uczenia

Pod redakcją
Ewy Skubalskiej-Rafajłowicz



Oficyna Wydawnicza Politechniki Wrocławskiej
Wrocław 2011

Recenzenci
Danuta RUTKOWSKA
Dariusz UCIŃSKI

Opracowanie redakcyjne i korekta
Hanna JUREK

Korekta
Alina KACZAK

Projekt okładki
Marcin ZAWADZKI

OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ
wyb. Stanisława Wyspiańskiego 27, 50-370 Wrocław
http://www.oficyna.pwr.wroc.pl; e-mail: oficwyd@pwr.wroc.pl
zamawianie.ksiazek@pwr.wroc.pl

ISBN 978-83-7493-603-3

Drukarnia Oficyny Wydawniczej Politechniki Wrocławskiej. Zam. nr 561/2011.

Spis treści

Spis najczęściej używanych oznaczeń i symboli

$f(.)$	–	funkcja aktywacji neuronu
$\mathbf{w}$	–	wektor wag sieci
$\mathbf{x} \in R^n$	–	wektor wejściowy
Ω	–	dziedzina
$\hat{\mathbf{y}}(\mathbf{x})$	–	wartość wyjścia z sieci odpowiadająca wektorowi wejściowemu $\mathbf{x}$
$\mathbf{y} \in R^m$	–	pożądany wektor wyjściowy
$(\mathbf{x}, \mathbf{y})$	–	element uczący
$\{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^N$	–	ciąg uczący
N	–	długość ciągu uczącego
$\mathbf{r}_k$	–	błąd na wyjściu sieci $\mathbf{r}_k = \mathbf{y}_k - \hat{\mathbf{y}}(\mathbf{x}_k)$
M	–	liczba neuronów w warstwie ukrytej
η	–	współczynnik uczenia
$\mathbf{H}$	–	macierz hesjanu
$\mathbf{G}$	–	macierz Greena
$\mathbf{z}$	–	wektor wejść kontekstowych
$\Psi(r)$	–	funkcja wpływu błędu
E	–	funkcja błędu (funkcja kryterialna)
Φ	–	funkcja radialna
$\mathbf{P}$	–	macierz rzutowania
λ	–	współczynnik regularyzacji
$\otimes$	–	iloczyn Kroneckera
t	–	czas

Rozdział 1

Wprowadzenie

Ewa Skubalska-Rafajłowicz

1.1. Przetwarzanie strumieni danych

Od pewnego czasu obserwuje się wzrost zainteresowania problemami przetwarzania dużych strumieni danych [2], [28], [44]. Polegają one na monitorowaniu wielowymiarowych szeregów czasowych – dużych strumieni danych – i podejmowaniu na ich podstawie decyzji.

Przykładami takich problemów jest:

– monitorowanie procesów produkcyjnych, w których skład coraz częściej wchodzą, poza konwencjonalnymi pomiarami parametrów procesu, obrazy z kamer przemysłowych [40], [52] czy innych typów czujników towarzyszących procesowi produkcyjnemu,

– monitorowanie danych z giełdy,

– monitorowanie ruchu pakietów w sieciach komputerowych [17], [59],

– analiza tekstów rejestrowanych w wersji elektronicznej [62],

– przetwarzanie danych w dużych bazach danych.

Zadania szczegółowe, które się w tych przypadkach pojawiają, to:

– analiza statystyczna monitorowanych danych, która poza tradycyjnym wyznaczaniem wartości średnich czy wariancji [7], [20], może polegać także na badaniu korelacji pomiędzy strumieniami (szeregami czasowymi) [64],

– analiza trendów [15], [50],

– grupowanie (klasteryzacja) danych [5], [8], [10], [12], [19], [30], [62],

– klasyfikacja aktualnych obserwacji [4], [41], [46], [61],

– adaptacyjne modelowanie procesu [35],

– wykrywanie zmian [3], [9], [18], [33] i wykrywanie nowości [45],

– wykrywanie obiektów,

– prognozowanie zachowania strumienia danych w bliższej i dalszej przyszłości i wiele innych specyficznie związanych z analizowanym procesem [13].

Ważną rolę ogrywa w wielu przypadkach redukcja wymiaru problemu [22], [51], [60]. Modele procesu mogą mieć charakter regresyjny [6], [42] lub polegać na bezpośrednim modelowaniu gęstości prawdopodobieństwa rozkładu danych [12], [23], [31], [32], [63].

Wszystkie te zadania można, a w wielu przypadkach powinno rozwiązywać się z użyciem sieci neuronowych; wymagają one jednak uwzględnienia specyfiki problemu związanej z koniecznością przetwarzania strumienia danych na bieżąco i towarzyszących temu ograniczeń czasowych i pamięciowych [29]. Potrzebne są szybkie algorytmy przetwarzające aktualne dane i metody gromadzenia informacji zawartej w przeszłych obserwacjach w możliwie skondensowany sposób [15], [24].

Niewątpliwie sieci neuronowe już w początkach swego rozwoju, na przełomie lat 50. i 60. XX wieku, były widziane jako modele, w których proces uczenia odbywał się na podstawie nieskończonych strumieni danych. W szczególności należy tu wymienić prace Widrowa [57], [58] w których pojawiają się związki algorytmów uczenia sieci nie tylko z metodami optymalizacyjmi, ale także z metodami aproksymacji stochastycznej [1], [21], [16], [38], [39], [53]. Na przykład algorytm wstecznej propagacji błędu używany do wyznaczania gradientu w algorytmach uczenia sieci jednokierunkowych mieści się w schemacie aproksymacji stochastycznej. W nurcie tym lokują się również algorytmy uczenia sieci samoorganizujących Kohonena [36], [37] czy też sieci służące do wyznaczania komponentów głównych [47], [48].

Algorytmy przeznaczone do przetwarzania dużych strumieni danych powinny mieć charakter przyrostowy, pozwalający modyfikować zgromadzoną na temat monitorowanego procesu wiedzę na podstawie aktualnych wartości strumienia danych [25], [26]. Sieć neuronowa powinna tu pełnić głównie rolę swego rodzaju pamięci. Dlatego pożądane jest, by proces uczenia sieci mógł przebiegać na bieżąco, równocześnie, i w związku z odbywającymi się procesami decyzyjnymi.

1.2. Zawartość tematyczna książki

Efektywne przetwarzanie nieograniczonych (i często wielowymiarowych) strumieni danych jest jednym z podstawowych zadań występujących w obecnej rzeczywistości, w szczególności w przypadku monitorowania, analizy i optymalizacji złożonych procesów przemysłowych. Pierwszy rozdział książki poświęcony jest metodom wspomagania procesów optymalizacyjnych w przypadku, gdy wyznaczanie wartości funkcji kryterialnej jest samo w sobie skomplikowaną procedurą, która odbywa się na podstawie dużego zbioru danych pomiarowych. Zbiór ten jest zależny nie tylko od położenia zmiennych decyzyjnych w przestrzeni poszukiwań, ale potencjalnie również zależnego od czasu. Problemy takie pojawiają się w zadaniach produkcyjnych optymalizowanych z użyciem metodologii przestrzeni odpowiedzi (ang. *Response Surface*) [11], [14], [34] lub sterowania wielobiegowego, w którym optymalizacja parametrów procesu następuje przed jego kolejnym

uruchomieniem na podstawie wiedzy zdobytej w poprzednich przebiegach oraz aktualnych dodatkowych pomiarach (ang. *run-to-run control*) [43], [49], [55].

Kryteriami optymalizacji mogą być tu nie tylko koszty produkcji, ale też stabilizacja własności produktu (minimalizacja zmienności, redukcja trendów). W rozdziale Marka Bazana „Sieci RBF w optymalizacji" sieć neuronowa tworzy zastępczy, lokalny model funkcji kryterialnej, którego użycie w procesie optymalizacji umożliwia zredukowanie liczby obliczeń wartości funkcji kryterialnej w obszarach, w których może być ona dostatecznie dokładnie aproksymowana. Do konstrukcji aproksymującej sieci z radialnymi funkcjami aktywacji RBF (ang. *Radial Basis Functions*) wykorzystano teorię samoreprodukujących funkcji jądrowych.

Rozdział autorstwa Mateusza Tykierki „Wykrywanie małych zmian w układach chaotycznych" obejmuje zagadnienia związane z modelowaniem chaotycznych strumieni danych za pomocą regularyzowanych sieci RBF oraz wykrywaniem zmian w tych strumieniach. Zakłada się tu, że dostępna informacja ograniczona jest do jednowymiarowego szeregu czasowego powstałego z próbkowania wyjść badanego systemu. Podane ograniczenia ułatwiają analizę rzeczywistych układów dynamicznych, o których informacje uzyskujemy na podstawie pomiarów eksperymentalnych. Jakość uzyskanego modelu została zweryfikowana z użyciem tak zwanych niezmienników dynamiki – wymiaru fraktalnego i maksymalnego wykładnika Lapunowa. Nawet przy wysokim poziomie szumu w zbiorze uczącym model uzyskany w pracy umożliwia poprawne odtworzenie dynamiki układu.

Ponieważ układy chaotyczne charakteryzują się krótkim horyzontem predykcji, po przekroczeniu którego dokładność modelowania załamuje się, autor zastosował cykliczną reinicjalizację modelu. W celu wykrywania małych zmian zachodzących w systemie zaproponowano sumowanie różnic pomiędzy wyjściem z systemu oraz generowanych przez sieć w ruchomym oknie czasowym o stałej długości. Autor pokazał, że można wykrywać nawet małe zmiany parametrów układu Lorenza (rzędu $0,02$) oraz sekwencje następujących po sobie zmian.

W rozdziale napisanym przez Andrzeja Rusieckiego „Odporne algorytmy uczenia sieci RBF", omówiono problem uczenia sieci neuronowych odpornego na błędy w danych uczących. Autor zaprezentował prosty, a zarazem szybki i efektywny algorytm pomagający poprawić odporność sieci o radialnych funkcjach bazowych. Algorytm ten opiera się na funkcji błędu w postaci sumy logarytmów kwadratów błędów LMLS (ang. *Least Mean Log Squares*).

Procedurę optymalizacyjną – proces uczenia sieci – przeprowadza się tu podobnie jak w przypadku często stosowanego algorytmu Levenberga–Marquardta. W metodzie tej, dzięki odpowiedniemu oszacowaniu macierzy hesjanu i zastosowaniu zmiennego w czasie czynnika regularyzacyjnego, w pobliżu rozwiązania hesjan funkcji kryterialnej aproksymowany jest jedynie za pomocą rozwinięcia pierwsze-

go rzędu, co prowadzi do przejścia do algorytmu Gaussa–Newtona o kwadratowej zbieżności, przy niezmienionym nakładzie obliczeniowym. W omawianym rozdziale pokazano inną metodę aproksymacji hesjanu, dopasowaną do odmiennej postaci kryterium (w stosunku do funkcji w postaci sumy kwadratów błędów) zastosowanej w przedstawionym odpornym algorytmie uczeni sieci RBF.

Zaletą opisanego tu odpornego algorytmu uczenia, w porównaniu z innymi algorytmami mającymi zapewniać odporność na błędy w danych, jest szybkość jego działania. Sposób działania algorytmu zilustrowano na przykładzie sterowania reaktorem przepływowym z mieszaniem, w którym przetwarzane dane nie mają wprawdzie wielu wymiarów, ale napływają w sposób ciągły. Zadaniem układu sterującego jest tu utrzymywanie stałego stężenia produktu reakcji, uzyskiwane przez odpowiednie dozowanie ilości jednego z dwóch substratów. Uwzględniono występowanie w systemie błędów pomiarowych oraz błędów grubych. Na podstawie danych pomiarowych opisana sieć RBF uczona jest przewidywania reakcji układu na sterowanie w danych warunkach.

Autorem kolejnego rozdziału noszącego tytuł „Uczenie ortogonalnych sieci neuronowych" jest Krzysztof Halawa. W rozdziale tym przedstawiono ortogonalne sieci neuronowe ONN (ang.*Orthogonal Neural Network*). Opisano sposoby nauki ONN ze szczególnym uwzględnieniem metod odpornych na błędy grube, które często stanowią poważny problem w wielu aspektach związanych z przetwarzaniem i akwizycją danych. Pokazano także algorytm szybkiego obliczania wyjść ONN z neuronami, których funkcje aktywacji należą do szeregu trygonometrycznego. W tym celu użyto efektywnych algorytmów wyznaczania dyskretnej transformaty Fouriera. Sprzętowa realizacja sieci neuronowych umożliwia w pełni wykorzystanie potencjału, jaki ma ich równoległa architektura. Układy programowalne FPGA (ang. *Field Programmable Gate Array*) doskonale nadają się do tworzenia sprzętowej implementacji ONN.

Piotr Ciskowski w rozdziale „Sieci kontekstowe i strumienie danych" opisał pewną klasę kontekstowych sieci neuronowych – model kontekstowego neuronu oraz struktury sieci kontekstowych, ich własności oraz algorytmy uczenia. Szczególnie dokładnie omówiono proces odwzorowania przestrzeni sygnałów wejściowych, zawierającej zmienne kontekstowe, w zmienne wyjściowe. Przeanalizowano przydatność sieci kontekstowych w przetwarzaniu intensywnych strumieni danych. Przedstawione przykłady działania sieci tradycyjnych oraz kontekstowych w zadaniach klasyfikacji oraz modelowania zależności finansowych. Sieć kontekstową zastosowano do wyceny opcji walutowych na giełdzie Forex. Jest to rynek o dużej płynności, w którym zmiany odbywają się w trybie ciągłym, przy czym notowania zmieniają się kilka razy na sekundę. Operacje wykonywane na tego typu danych mogą obejmować zarówno używanie sieci neuronowych do wyce-

ny opcji, bądź szacowania zmienności rynku, jak i ciągłe douczanie sieci w celu dostosowania ich do zmieniających się warunków rynkowych.

Prace zawarte w niniejszej monografii były finansowane przez Ministerstwo Nauki i Szkolnictwa Wyższego w ramach grantu badawczego w latach 2006–2009.

Bibliografia

[1] Alessandri A., Parisini T., *Nonlinear Modelling of Complex Large-Scale Plants Using Neural Networks and Stochastic Approximation*, IEEE Trans. Syst., Man, Cybernetics A 27(6): 750–757, 1997.

[2] Aggarwal C., *Data Streams: Models and Algorithms*, Springer, 2007.

[3] Aggarwal C., *A Framework for Diagnosing Changes in Evolving Data Streams*, Proc. of the ACM SIGMOD Conference, 2003.

[4] Aggarwal C., Han J., Wang J., Yu P.S., *On Demand Classification of Data Streams*, Proc. of Int. Conf. on Knowledge Discovery and Data Mining (KDD '04), Seattle, WA, Aug. 2004.

[5] Aggarwal C., Han J., Wang J., Yu P.S., *A Framework for Projected Clustering of High Dimensional Data Streams*, Proc. of Int. Conf. on Very Large Data Bases (VLDB '04), Toronto, Canada, Aug. 2004.

[6] Babcock B., Babu S., Datar M., Motwani R., Widom J., *Models and issues in data stream systems*, Proc. of the 21^{st} ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002):1–16, 2002.

[7] Babcock B., Datar M., Motwani R., O'Callaghan L., *Maintaining Variance and k-Medians over Data Stream Windows*, Proc. of the 22^{nd} Symposium on Principles of Database Systems (PODS 2003): 234–243, 2003.

[8] Banerjee A., Ghosh J., *Frequency-Sensitive Competitive Learning for Scalable Balanced Clustering on High-Dimensional Hyperspheres*, IEEE Tran. on Neural Networks 15(3): 702–719, 2004.

[9] Ben-David S., Gehrke J., Kifer D., *Detecting Change in Data Streams*, Proc. of Int. Conf. on Very Large Data Bases (VLDB '04), Toronto, Canada, Aug. 2004.

[10] Beringer J., Hullermeier E., *Online Clustering of Parallel Data Streams*, Data and Knowledge Engineering, 2005.

[11] Box G., Draper N., *Empirical model-building and response surfaces* New York: John Wiley, 1987.

[12] Cao F., Ester M., Qian W., Zhou A., *Density-based Clustering over an Evolving Data Stream with Noise*, Proc. of the 2006 SIAM Conference on Data Mining (SDM '2006), 2006.

[13] Chakrabarti A., Do Ba K., Muthukrishnan S., *Estimating entropy and entropy norm on data streams* Internet Math. 3(1): 63–78, 2006.

[14] Chatterjee S.K., Mandal N.K., *Response surface designs for estimating the optimal point*, Calcutta Statist. Assoc. Bull. 30: 145–169, 1981.

[15] Chen Y., Dong G., Han J., Wah B.W., Wang J., *Multi-Dimensional Regression*

Analysis of Time-Series Data Streams, Proc. of Int. Conf. on Very Large Data Bases (VLDB '02), 2002.

[16] Cherkassky V., Mulier F., *Learning from data: concepts, theory, and methods*, John Wiley and Sons, Inc., Hoboken, New Jersey 2007.

[17] Cormode G., Muthukrishnan S., *What is new: Finding significant differences in network data streams*, INFOCOM 2004.

[18] Chu F., Zaniolo C., *Fast and light boosting for adaptive mining of data streams*, Proc. of the 5^{th} Pacific-Asic Conference on Knowledge Discovery and Data Mining (PAKDD), Sydney, 2004.

[19] O'Callaghan L., Mishra N., Meyerson A., Sudipto G., Motwani R., *Streaming-data algorithms for high-quality clustering*, Proc. of IEEE International Conference on Data Engineering, 2002.

[20] Datar M., Gionis A., Indyk P., Motwani R., *Maintaining Stream Statistics Over Sliding Windows* , Proc. of 13^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002), 2002.

[21] Du K.L., Swamy M.N.S., *Neural Networks in a Softcomputing framework*, Springer-Verlag, London 2006.

[22] Egecioglu O., Ferhatosmanoglu H., Ogras U., *Dimensionality Reduction and Similarity Computation by Inner-Product Approximations*, IEEE Tran. on Knowledge and data Engineering 16(6): 714–726, 2004.

[23] Kerdprasop N., Kerdprasop K., *Density Estimation Technique for Data Stream Classification*, 17^{th} International Conference on Database and Expert Systems Applications (DEXA '06): 662–666, 2006.

[24] Fan W., *Systematic data selection to mine concept-drifting data streams*, KDD 2004: 128–137, 2004.

[25] Ferrer-Troyano F.J., Aguilar-Ruiz J.S., Riquelme J.C., *Discovering Decision Rules from Numerical Data Streams*, ACM Symposium on Applied Computing - SAC '04: 649–653, 2004.

[26] Ferrer-Troyano F.J., Aguilar-Ruiz J.S., Riquelme J.C., *Incremental Rule Learning based on Example Nearness from Numerical Data Streams*, ACM Symposium on Applied Computing – SAC '05: 568–572, 2005.

[27] Gaber M.M., Zaslavsky A., Krishnaswamy S., *Mining Data Streams: A Review*, SIGMOD Record 34(2): 18–26, 2005.

[28] Gao J., Fan W., Han J., *On Appropriate Assumptions to Mine Data Streams: Analysis and Practice*, IEEE International Conference on Data Mining (ICDM'07), Omaha, NE, 2007.

[29] Gilbert A.C., Kotidis Y., Muthukrishnan S., Strauss M., *One-Pass Wavelet Decompositions of Data Streams*, TKDE 15(3): 541–554, 2003.

[30] Guha S., Meyerson A., Mishra N., Motwani R., O'Callaghan L., *Clustering Data Streams: Theory and Practice*, IEEE Tran. on Knowledge and data Engineering 15(3): 515–528, 2003.

[31] Heinz C., Seeger B., *Towards Kernel Density Estimation over Streaming Data*, Proc. of COMAD, Delhi, India, 2006.

[32] Heinz C., Seeger B., *Stream Mining via Density Estimators: A concrete Application*, Proc. of COMAD, Delhi, India, 2006.

[33] Ho S.S., *A martingale framework for concept change detection in time-varying data streams*, Proc. of the 22^{nd} International Conference on Machine Learning, Bonn, Germany: 321–327, 2005.

[34] Hoyte D.W., Liptak B.G., *Empirical Process Optimization*, Instrument Engineers' Handbook (4 wyd.) Process Control and Optimization II, CRC Press Taylor and Francis Group, Boca Raton, FL, 157–161, 2006.

[35] Jin C., Qian W., Sha C., Yu J.X., Zhou A., *Dynamically Maintaining Frequent Items over a Data Stream*, Proc. of the 12^{th} ACM Conference on Information and Knowledge Management (CIKM '2003), 2003.

[36] Kohonen T., *Self-organized formation of topologically correct feature maps*, Biolog. Cybernetics 43: 59–69, 1982.

[37] Kohonen T., *Self-organization and Associative Memory*, New York, Springer-Verlag, 2 wyd., 1988.

[38] Koronacki J., *Aproksymacja stochastyczna: metody optymalizacji w warunkach losowych*, WNT, Warszawa, 1989.

[39] Kushner H.J., Yin G.G., *Stochastic Approximation and Recursive Algorithms and Applications*, Springer-Verlag New York, Inc., 2003.

[40] Laerhoven K.V., Aidoo K., Lowette S., *Real-time Analysis of Data from Many Sensors with Neural Networks*, Proc. of the fourth International Symposium on Wearable Computers (ISWC) Zurich, 7–9 October 2001.

[41] Last M., *Online Classification of Nonstationary Data Streams*, Intelligent Data Analysis 6(2): 129–147, 2002.

[42] Law M.H.C., Zhang N., Jain A., *Nonlinear Manifold Learning for Data Stream*, Proc. of SIAM International Conference on Data Mining, 2004.

[43] Moyne J., del Castillo Arnon E., Hurwitz M., *Run-to-Run Control in Semiconductor Manufacturing*, CRC Press LLC, Boca Raton, Florida, 2001.

[44] Muthukrishnan S., *Data Streams: Algorithms and Applications*, Foundations and Trends in Theoretical Computer Science 1(2), 2005.

[45] Muthukrishnan S., Shah R., Vitter J.S., *Mining Deviants in Time Series Data Streams*, Proc. of 16^{th} International Conference on Scientific and Statistical Database Management (SSDBM '04), 2004.

[46] Nasraoui O., Rojas C., *Robust Clustering for Tracking Noisy Evolving Data Streams*, Proc. SIAM conference on Data Mining, Bethesda, 2006.

[47] Oja E., *A simplified neuron model as a principal component analyzer*, Journal of Mathematical Biology, 15:267–273, 1982.

[48] Oja E., Karhunen J., *On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix*, Journal of Mathematical Analysis and Applications 106: 69–84, 1985.

[49] Patel N.S., Jenkins S.T., *Adaptive Optimization of Run-to-Run Controllers: The EWMA Example*, IEEE Tran on Semiconductor Engineering 13(1): 97–107, 2000.

[50] Rajaraman K., Tan A.H., *Topic Detection, Tracking, and Trend Analysis Using Self-Organizing Neural Networks*, Proc. of PAKDD 2001: 102–107, 2001.

[51] Skubalska-Rafajlowicz E., *Random Projection RBF Nets for Multidimensional Density Estimation*, Int. J. Appl. Math. Comput. Science 18(4): 1–10, 2008.

[52] Skubalska-Rafajlowicz E., *Local correlation and entropy maps as tools for detecting defects in industrial images*, Int. J. Appl. Math. Comput. Science 18(1): 41–47, 2008.
[53] Spall J.C., Cristion J.A., *Direct Adaptive Control of Nonlinear Systems Using Neural Networks and Stochastic Approximation*, Proc. of 31 IEEE Conf Decision Control, (Tucson, AZ): 878–883, 1992.
[54] Tadeusiewicz R., *Sieci neuronowe*, Akademicka Oficyna Wydawnicza, Warszawa, 1993.
[55] Wang C.C., Jiang B.C., Wub M.Y., Jen C.H., *Evaluation of fuzzy neural network run-to-run controller using numerical simulation analysis for SISO process*, Expert Systems with Applications 36: 12044–12048, 2009.
[56] White H., *Some Asymptotic Results for Learning in Single Hidden-Layer Feed forward Network Models*, Journal of the American Statistical Association 84(408): 1003–1014, 1989.
[57] Widrow B., Lehr M.A., *30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation*, Proc. of the IEEE 78(9): 1415–1442, 1990.
[58] Widrow B., Kamenetsky M., *Statistical efficiency of adaptive algorithms*, Neural Networks 16: 735–744, 2003.
[59] Yoshida K., Katsuno S., Ano S., Yamazaki K., Masato Tsuru M., *Stream Mining for Network Management*, IEICE Trans Commun. E89-B(6): 1774–1780, 2006.
[60] Yue X., Mob H., Chi Z.X., *Immune-inspired incremental feature selection technology to data streams*, Applied Soft Computing 8: 1041–1049, 2008.
[61] Zaki M.J., Aggarwal C.C., *XRules: An Effective Structural Classifier for XML Data*, Proc. of SIGKDD '03, 2003.
[62] Zhong S., *Efficient streaming text clustering*, Neural Networks 18: 790–798, 2005.
[63] Zhou A., Cai Z., Wei L., Qian W., *M-kernel merging: towards density estimation over data streams*, Proc. of Database Systems for Advanced Applications, 2003 (DASFAA 2003): 285–292, 2003.
[64] Zhu Y., Shasha D., *StatStream: Statistical monitoring of thousands of data streams in real time*, Proc. of the 28 VLDB Conf., Hong Kong, China: 358–369, 2002.

Rozdział 2

Sieci RBF w optymalizacji

Marek Bazan

2.1. Wprowadzenie

Procesy optymalizacyjne, w których wyznaczanie wartości funkcji celu jest bardzo czasochłonne, występują przede wszystkim jako element procesów konstrukcyjnych lub optymalizacyjnych różnego rodzaju urządzeń technicznych. Wyznaczanie wartości funkcji, które jest kosztowne, pojawia się również w zagadnieniach, w których, aby znaleźć optymalne wartości parametrów, należy wykonać serię eksperymentów. Algorytmy umożliwiające skonstruowanie aproksymacji funkcji celu w takich procesach, a przez to przyspieszenie ich zbieżności lub mogące posłużyć jako narzędzie eksploracji przestrzeni rozwiązań, są pożądane.

Podstawowymi cechami, jakie musi mieć metoda aproksymacji użyta w takim algorytmie, jest dobra jakość uogólniania rozwiązania oraz szybkość konstrukcji. Cechy takie mają sieci z radialnymi funkcjami bazowymi. W niniejszym rodziale prezentujemy zastosowania sieci z radialnymi funkcjami bazowymi w procesach optymalizacyjnych z funkcjami celu, których wyznaczanie wartości jest czasochłonne.

Rozdział podzielony jest na dwie części. W pierwszej części omawiamy problemy interpolacji i aproksymacji funkcji n zmiennych za pomocą sieci z radialnymi funkcjami aktywacji. Jako metodę rozwiązania zagadnienia aproksymacyjnego omawiamy metodę regularyzacji Tichonowa z teorii zagadnień źle postawionych [14], [22], [28]. Podajemy charakterystykę przestrzeni funkcji, które mogą być aproksymowane sieciami z radialnymi funkcjami bazowymi [30]. Następnie cytujemy znane twierdzenia wraz ze szkicami dowodów o ograniczeniach błędów interpolacji oraz aproksymacji, które zachodzą, gdy założymy, że zbiór danych zawiera dużo danych równomiernie rozłożonych [16], [24], [31], [32]. Następnie przytaczamy analogiczne twierdzenie wykorzystujące funkcję wzrostu wielomianów [12], które nie wymaga założenia o gęstości zbioru danych. Dla kompletności omawiamy wskaźniki mierzące gęstość zbioru danych. Następnie omawiamy trzy różne metody wyboru parametru regularyzacji w rozwiązaniu zadania

aproksymacyjnego. Są to: uogólniona walidacja krzyżowa GCV (ang. *Generalized Cross-Validation*) [29], metoda L–krzywej (ang. *L–curve*) [14] oraz metoda zaproponowana przez autora i in. w [4]. Podajemy wyniki numerycznych testów dla metod GCV, L–krzywej oraz metody zaproponowanej przez autora dla zbiorów danych pochodzących z optymalizacji funkcji testowej bezgradientowym algorytmem zaproponowanym w [15] i używanym do optymalizacji magnesów nadprzewodzących [23].

W drugiej części pracy przedstawiamy dwa podejścia do przyspieszania procesów optymalizacyjnych przez konstrukcję aproksymacji funkcji celu. Na podstawie [8] opisujemy metodę regionu wiarygodności oraz jej realizację z użyciem sieci z radialnymi funkcjami bazowymi zaproponowaną w [21]. Drugie z omawianych podejść, zaproponowane przez autora i in. w [4], polega na połączeniu algorytmu optymalizacyjnego z konstrukcją zregularyzowanej sieci neuronowej z funkcjami Gaussa i obliczeniem za jej pomocą aproksymacji wartości funkcji celu w punktach generowanych w regionach dostatecznie bogatych w punkty dane, dla których wartość funkcji celu została obliczona w sposób bezpośredni. Pokazujemy wyniki zastosowania tej metody do optymalizacji funkcji testowej Rosenbrocka oraz opisujemy wyniki rzeczywistego zastosowania w procesie optymalizacji magnesów zakrzywiających dla Wielkiego Zderzacza Hadronów zbudowanego CERNie [5], [23]. Warunki zbieżności i dowód zbieżności zaproponowanej przez autora metody zaprezentowane zostały w pracy [2] (również zob. [1]).

2.2. Aproksymacja funkcji

2.2.1. Zagadnienie interpolacji a zagadnienie aproksymacji

Zajmiemy się dwoma typami zagadnień – interpolacją oraz aproksymacją funkcji ciągłych n zmiennych. W obydwu zagadnieniach idzie o przybliżenie funkcji ciągłej $g\colon \mathbb{R}^n \to \mathbb{R}$, która zadana jest za pomocą zbioru dyskretnego $\mathbf{Z} = \{\mathbf{x}_i, y_i\}_{i=1}^N$. W zagadnieniu interpolacji zbiór $\mathbf{Z}$ jest taki, że

$$g(\mathbf{x}_i) = y_i, \quad i = 1, \ldots, N. \tag{2.1}$$

Natomiast w wypadku aproksymacji dopuszczamy zaburzenie wartości funkcji g na poziomie ϵ, tzn.:

$$g(\mathbf{x}_i) = y_i + \delta_i \quad \text{i} \quad |\delta_i| \leqslant \epsilon, \quad i = 1, \ldots, N. \tag{2.2}$$

Rozwiązania podanych zagadnień poszukujemy w zbiorze funkcji postaci

$$s_{g,Z}(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(||\mathbf{x} - \mathbf{x}_i||) + \sum_{j=1}^{Q} b_j p_j(\mathbf{x}), \tag{2.3}$$

gdzie ϕ jest bazową funkcją radialną, natomiast $\{p_j\}_{j=1}^{Q}, Q = \binom{m+n-1}{n}$ jest bazą przestrzeni wielomianów n zmiennych stopnia nie wyższego niż m.

Warunki istnienia oraz jednoznaczności rozwiązania zagadnienie interpolacji sprowadzają się do odpowiedzi na pytanie o odwracalność macierzy układu

$$\begin{pmatrix} A & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix}, \tag{2.4}$$

gdzie $A = (\phi(||\mathbf{x}_i - \mathbf{x}_j||))_{i,j=1,\ldots,N}$ i $P = (p_j(\mathbf{x}_i))_{i=1,\ldots,N,j=0,\ldots,Q}$ oraz $\mathbf{w} = (w_1, w_2, \ldots, w_N)^T$, $\mathbf{b} = (b_1, b_2, \ldots, b_Q)^T$ i $\mathbf{y} = (y_1, y_2, \ldots, y_N)$. Zauważmy, że rozmiar dolnej część układu (2.4), tzn. $(P^T\ 0)$, określony jest przez stopień m części wielomianowej w funkcji interpolującej (2.3). Stopień ten zależy od tego, jaka radialna funkcja bazowa ϕ została użyta. Jeśli ϕ jest funkcją ściśle dodatnio określoną (zob. paragraf 2.2.2), mamy $m = 0$ i wówczas macierz układu (2.4) sprowadza się do macierzy A. Jeśli ϕ jest funkcją warunkowo dodatnio określoną (zob. paragraf 2.2.2), to dolna część układu określona jest przez warunek (2.9).

Warunki, jakie musi spełniać zbiór $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ i radialna funkcja bazowa ϕ, aby macierz układu (2.9) była nieosobliwa, po raz pierwszy zostały podane w [19].

Metodą rozwiązania zagadnienia aproksymacyjnego, jaką zajmiemy się w niniejszym rozdziale, jest metoda regularyzacji Tichonowa. Jej korzenie sięgają prac nad problemami źle postawionymi (zob. [28] oraz referencje tamże). Zregularyzowane rozwiązanie zagadnienia aproksymacyjnego jest poszukiwane w przestrzeni Hilberta $\mathcal{H}$. Operator ograniczeń jest definiowany za pomocą innej przestrzeni unormowanej $\mathcal{G}$, dla której określony jest liniowy operator $L\colon \mathcal{H} \to \mathcal{G}$. Operator ograniczeń $J\colon \mathcal{H} \to \mathbb{R}$ określony jest wówczas jako $J(s) = ||Ls||_{\mathcal{G}}^2$. Zagadnienie regularyzacji polega na znalezieniu dla ustalonej wartości parametru $\lambda > 0$, funkcji $s^* \in \mathcal{H}$ postaci (2.3), która jest rozwiązaniem zagadnienia minimalizacyjnego

$$\min_{s \in \mathcal{H}} \left\{ \sum_{j=1}^{N} [g(\mathbf{x}_j) - s(\mathbf{x}_j)]^2 + \lambda J(s) \right\}. \tag{2.5}$$

Jeśli przyjmiemy, że $J(s) = ||s||_{\mathcal{G}}^2$, to zagadnienie sprowadza się do rozwiązania układu

$$\begin{pmatrix} A + \lambda I & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix} \tag{2.6}$$

dla ustalonej wartości parametru $\lambda > 0$. Warunki istnienia i jednoznaczności rozwiązania tego zagadnienia podała Wahba [29]. Oczywiście kluczowym zagadnieniem jest wybór wartości parametru λ. Omówione zostaną tu trzy metody wyboru wartości tego parametru. Stabilne metody rozwiązania układu (2.6) znaleźć można w monografii [14]. Dla stosunkowo małego układu równań, takiego z jakim mamy do czynienia w aproksymacji funkcji celu w procesach optymalizacyjnych, do rozwiązania możliwe jest użycie różnych wariantów rozkładu SVD.

2.2.2. Radialne funkcje bazowe

Niech $\pi_m(\mathbb{R}^n)$ oznacza przestrzeń wielomianów n zmiennych stopnia nie większego niż m. Rozróżniamy radialne funkcje bazowe dodatnio określone oraz warunkowo dodatnio określone. Definicję obu klas podajemy za [7].

Definicja 1
Mówimy, że funkcja $\phi : \mathbb{R} \to \mathbb{R}$ jest całkowicie monotoniczna, jeśli

$$(-1)^r \phi^{(r)} \geqslant 0, \qquad r = 1, 2, \ldots \tag{2.7}$$

Twierdzenie 1 (zob. [7])
Zakładając, że punkty w zbiorze $\mathbf{X}$ są różne, macierz A jest dodatnio określona, jeśli funkcja ϕ z centrum w $\mathbf{c} \in \mathbb{R}^n$ jest taka, że dla $r = ||\mathbf{x} - \mathbf{c}||$ funkcja $\phi(\sqrt{r})$ jest całkowicie monotoniczna i nie jest stała.

Definicja 2
Mówimy, że funkcja $\phi : \mathbb{R}^n \to \mathbb{R}$ jest warunkowo dodatnio określona stopnia k, jeśli dla każdego zbioru punktów $\mathbf{X} \subset \mathbb{R}^n$ forma kwadratowa

$$\sum_{i=1}^{N} \sum_{k=1}^{N} w_i w_k \phi(\mathbf{x}_i - \mathbf{x}_k) \tag{2.8}$$

jest nieujemna dla wszystkich niezerowych $\{w_i\}_{i=1,\ldots,N}$, które spełniają

$$\sum_{i=1}^{N} w_i p(\mathbf{x}_i) = 0, \tag{2.9}$$

gdzie $p(\mathbf{x})$ jest dowolnym wielomianem n zmiennych stopnia nie większego niż m. Mówimy, że funkcja ϕ jest ściśle warunkowo dodatnio określona stopnia m, jeśli podana forma kwadratowa jest zawsze dodatnia.

Twierdzenie 2 (zob. [7])
Dla funkcji ϕ, które są warunkowo dodatnio określone stopnia m, funkcja

$$(-1)^k \phi^{(k)}(\sqrt{t}) \tag{2.10}$$

jest całkowicie monotoniczna.

Najczęściej używanymi w praktyce funkcjami radialnymi w kontekście zagadnień interpolacji i aproksymacji funkcji są: spośród funkcji ściśle dodatnio określonych, tzn. takich, dla których $m = 0$ (zob. [27], [30]), funkcje

$$\begin{aligned} \phi(r) &= (r^2 + c^2)^\beta, \quad \beta < 0, \\ \phi(r) &= e^{-\alpha r^2}, \quad \alpha > 0, \\ \phi(r) &= (1-r)_+^4(4r+1) \end{aligned}$$

oraz spośród warunkowo dodatnio określonych stopnia m funkcje

$$\begin{aligned} \phi(r) &= (-1)^{\beta/2+1} r^\beta \log r, \quad \beta \in 2\mathbb{N}, && \text{gdzie} \quad m = \beta/2 + 1, \\ \phi(r) &= r^\beta \quad \beta \in \mathbb{R}_{>0}/2\mathbb{N}, && \text{gdzie} \quad m = \lceil \beta/2 \rceil + 1, \\ \phi(r) &= (-1)^{\lceil \beta \rceil} (r^2 + c^2)^\beta, \quad \beta > 0, && \text{gdzie} \quad m = \lceil \beta \rceil. \end{aligned}$$

Cechą charakterystyczną funkcji ściśle dodatnio określonych jest ich wykres w kształcie „dzwonu" (ang. *bell-shaped*) oraz $\lim_{r\to\infty} \phi(r) = 0$. Dla funkcji warunkowo dodatnio określonych, czyli takich, dla których $m > 0$, mamy natomiast $\lim_{r\to\infty} \phi(r) \to \infty$.

Dla funkcji warunkowo dodatnio określonych, aby macierz układu (2.6) była macierzą nieosobliwą, wymagane jest, aby zbiór $\mathbf{X}$ spełniał warunek unisolwentności względem przestrzeni $\pi_m(\mathbb{R}^n)$.

Definicja 3 (unisolwentność zbioru $\mathbf{X}$)
Mówimy, że zbiór punktów $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subseteq \mathbb{R}^n$, gdzie $N > Q = \dim \pi_m(\mathbb{R}^n)$ jest unisolwentny względem przestrzeni $\pi_m(\mathbb{R}^n)$, jeśli wielomian zerowy jest jedynym wielomianem z przestrzeni $\pi_m(\mathbb{R}^n)$, który znika na wszystkich punktach $\mathbf{X}$ jednocześnie.

Własność unisolwentności zbióru $\mathbf{X}$ gwarantuje nam, że zbiór ten próbkuje przestrzeń $\mathbb{R}^n$ w każdym wymiarze w taki sposób, że możliwe jest skonstruowanie wielomianu interpolacyjnego n zmiennych pełnego stopnia m. Przykładem zbioru, który nie jest unisolwenty względem $\pi_2(\mathbb{R}^n)$, jest zbiór 6 punktów położonych na okręgu. Na takim zbiorze nie można zbudować kwadratowego wielomianu interpolacyjnego.

2.2.3. Przestrzeń funkcji, które można aproksymować

Przestrzeń funkcji, które można aproksymować za pomocą rozwinięcia (2.3) dla ustalonej dodatnio określonej funkcji radialnej $\Phi(\mathbf{x}, \mathbf{y}) = \phi(||\mathbf{x} - \mathbf{y}||_2)$ jest

przestrzenią Hilberta $\mathcal{H}$ z ustalonym iloczynem skalarnym $(\cdot,\cdot)_{\mathcal{H}}$, zawierającą samoreprodukującą funkcję jądrową. Ograniczenia błędu aproksymacji, jakimi się zajmiemy w następnym podrozdziale, są postaci

$$|g - s_{g,\mathbf{Z}}| \leqslant CF(h(x))||g||_{\mathcal{H}}$$

gdzie norma $||\cdot||_{\mathcal{H}}$ jest normą generowaną przez iloczyn skalarny $(\cdot,\cdot)_{\mathcal{H}}$, $C > 0$ jest stałą, $h(\mathbf{x})$ jest miarą gęstości danych, a F jest funkcją ciągłą.

Niniejszy podrozdział został opracowany na podstawie [25], [27].

Przestrzeń Hilberta z samoreprodukującą funkcją jądrową

Niech $\Omega \subseteq \mathbb{R}^n$ będzie dziedziną funkcji rzeczywistych, tworzących przestrzeń Hilberta $\mathcal{H}$ z iloczynem skalarnym $(\cdot,\cdot)_{\mathcal{H}}$. Załóżmy ponadto, że dla każdego $\mathbf{x} \in \Omega$ funkcjonał wyznaczania wartości w punkcie $\delta_{\mathbf{x}} : g \to g(\mathbf{x})$ jest ciągły w $\mathcal{H}$, tzn.

$$\delta_{\mathbf{x}} \in \mathcal{H}^* \qquad \text{dla każdego } \mathbf{x} \in \Omega,$$

gdzie $\mathcal{H}^*$ jest przestrzenią dualną do $\mathcal{H}$, tzn. przestrzenią ciągłych funkcjonałów liniowych określonych na $\mathcal{H}$.

Twierdzenie 3
Jeśli przestrzeń Hilberta funkcji określonych na zbiorze Ω zawiera ciągły funkcjonał wyznaczania wartości w punkcie, to zawiera symetryczną funkcję nazywaną reprodukującą funkcją jądrową $\Phi : \Omega \times \Omega \to \mathbb{R}$ o własnościach

$$\begin{aligned}
\Phi(\mathbf{x},\cdot) &\in \mathcal{H},\\
g(\mathbf{x}) &= (g, \Phi(\mathbf{x},\cdot))_{\mathcal{H}},\\
\Phi(\mathbf{x},\mathbf{y}) &= (\Phi(\mathbf{x},\cdot), \Phi(\mathbf{y},\cdot))_{\mathcal{H}} = \Phi(\mathbf{y},\mathbf{x}),\\
\Phi(\mathbf{x},\mathbf{y}) &= (\delta_{\mathbf{x}}, \delta_{\mathbf{y}})_{\mathcal{H}^*} = \Phi(\mathbf{y},\mathbf{x}),
\end{aligned}$$

dla wszystkich $\mathbf{x},\mathbf{y} \in \Omega$, $g \in \mathcal{H}$.

Przestrzeń natywna

Definicja 4
Jeśli symetryczna (warunkowo) dodatnio określona funkcja $\Phi : \Omega \times \Omega \to \mathbb{R}$ jest reprodukującą funkcją jądrową w przestrzeni Hilberta $\mathcal{H}$ funkcji o wartościach rzeczywistych określonych na zbiorze Ω, to przestrzeń $\mathcal{H}$ nazywamy przestrzenią natywną generowaną przez funkcję jądrową Φ i oznaczamy ją $\mathcal{N}_{\Phi}(\mathbb{R}^n)$.

Jednoznaczność przestrzeni natywnej generowanej przez funkcję Φ opisywana jest następującym twierdzeniem.

Twierdzenie 4
Przestrzeń natywna $\mathcal{H} = \mathcal{N}_\Phi(\mathbb{R}^n)$ generowana przez daną (warunkowo) dodatnio określoną funkcję Φ, o ile istnieje, jest określona jednoznacznie. Ponadto pokrywa się z domknięciem przestrzeni skończonych kombinacji liniowych funkcji $\Phi(\mathbf{x}, \cdot)$ względem iloczynu skalarnego

$$(\Phi(\mathbf{x}, \cdot), \Phi(\mathbf{y}, \cdot))_\mathcal{H} = \Phi(\mathbf{x}, \mathbf{y}) \qquad \textit{dla każdego } \mathbf{x}, \mathbf{y} \in \Omega.$$

Według twierdzenia 4 przestrzeń natywna generowana przez radialną funkcję bazową Φ jest domknięciem przestrzeni składającej się z funkcji postaci

$$g_\mathbf{w} := \sum_{j=1}^{N} w_j \Phi(\cdot - \mathbf{x}_j), \quad w_j \in \mathbb{R}, \quad j = 1, \ldots, N.$$

Iloczyn skalarny w tej przestrzeni zdefiniowany jest jako

$$(\cdot, \cdot)_\mathcal{H} = (g_{\mathbf{w}^{(1)}}, g_{\mathbf{w}^{(2)}})_\Phi := \sum_{j=1}^{N} \sum_{i=1}^{M} w_j^{(1)} w_i^{(2)} \Phi(\mathbf{y}_i - \mathbf{x}_j), \tag{2.11}$$

gdzie $g_{\mathbf{w}^{(1)}}$ jest funkcją interpolującą funkcję g na zbiorze $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, natomiast $g_{\mathbf{w}^{(2)}}$ jest funkcją interpolującą funkcję g na zbiorze $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$. W dowodach twierdzeń o ograniczeniu błędu interpolacji i aproksymacji, które podamy w następnym podrozdziale, potrzebne jest ponadto założenie o funkcji g, że jej transformata Fouriera $\hat{f}$ jest zdominowana przez transformatę Fouriera $\hat{\Phi}$ funkcji $\Phi(r) = \phi(||r||_2)$ w sensie

$$\int |\hat{f}|^2 \hat{\Phi}^{-1} dt < \infty. \tag{2.12}$$

Założenie to oznacza, że iloczyn skalarny (2.11) można zapisać w postaci całki Lebesgue'a

$$\begin{aligned}(\cdot, \cdot)_\mathcal{H} = \; & (g_{\mathbf{w}^{(1)}}, g_{\mathbf{w}^{(2)}})_\Phi := \sum_{j=1}^{N} \sum_{i=1}^{M} w_j^{(1)} w_i^{(2)} \Phi(\mathbf{y}_i - \mathbf{x}_j) \\ = \; & (2\pi)^{-n} \int_{\mathbb{R}^n} \hat{\Phi}(\omega) \left(\sum_{j=1}^{N} w_j^{(1)} e^{i\mathbf{x}_j^T \omega} \right) \left(\sum_{k=1}^{M} w_k^{(2)} e^{-i\mathbf{y}_k^T \omega} \right) d\omega.\end{aligned}$$

Mając daną dodatnio określoną funkcję Φ, interesuje nas pytanie, jakie funkcje należą do przestrzeni natywnej generowanej przez funkcję Φ. Ogólną odpowiedź

można znaleźć w [27]. Ograniczymy się do przestrzeni zawierających funkcje, które mogą być funkcjami celu w procesie optymalizacyjnym. Aby można było użyć algorytmu bezgradientowego, funkcja celu musi być co najmniej ciągła. Algorytmy gradientowe lub drugiego rzędu wymagają odpowiednio ciągłości pochodnych kierunkowych oraz cząstkowych drugiego rzędu.

Dalej omówimy więc przestrzeń natywną, która zawiera funkcje o pochodnych całkowalnych z pewną potęgą. Przestrzeń taka jest przestrzenią Sobolewa. Pokażemy, jakie dodatnio określone funkcje generują przestrzenie natywne izomorficzne z przestrzeniami Sobolewa odpowiedniego rzędu. Przestrzenie Sobolewa odpowiednich rzędów zawierają przestrzenie $C(\mathbb{R}^n)$, $C^1(\mathbb{R}^n)$ i $C^2(\mathbb{R}^n)$. Na koniec zajmiemy się przestrzenią natywną generowaną prze funkcje Gaussa. Jest to przestrzeń często używana w zastosowaniach praktycznych.

Przestrzenie Sobolewa

Przestrzeń Sobolewa definiujemy za pomocą operatora różniczkowania D^α

$$D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \dots \partial x_n^{\alpha_n}}, \tag{2.13}$$

gdzie α jest wielowskaźnikiem $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ oraz $\alpha_i \in \mathbb{N}$. Przyjmuje się ponadto oznaczenie $|\alpha| = \alpha_1 + \alpha_2 + \dots + \alpha_n$.

Definicja 5 (przestrzeń Sobolewa całkowitego rzędu)
Przestrzeń Sobolewa $W_p^k(\Omega)$ określamy jako przestrzeń funkcji $u : \Omega \to \mathbb{R}$, dla których $D^\alpha u \in L_p(\Omega), |\alpha| \leqslant k$. Jest to (semi-)przestrzeń Hilberta z (semi-)normą określoną odpowiednio

$$|u|_{W_p^k(\Omega)} := \left(\sum_{|\alpha|=k} ||D^\alpha u||^p_{L_p(\Omega)} \right)^{1/p} \quad oraz \quad ||u||_{W_p^k(\Omega)} := \left(\sum_{|\alpha|\leqslant k} ||D^\alpha u||^p_{L_p(\Omega)} \right)^{1/p}.$$

Definicja 6 (przestrzeń Sobolewa ułamkowego rzędu)
Przestrzeń Sobolewa $W_p^{k+s}(\Omega), 1 \leqslant p < \infty, k \in \mathbf{N}_0, 0 < s < 1$ określamy jako przestrzeń funkcji $u\colon \Omega \to \mathbb{R}$, dla których poniższe normy są skończone

$$|u|_{W_p^{k+s}(\Omega)} := \left(\sum_{|\alpha|=k} \int_\Omega \int_\Omega \frac{|D^\alpha u(\mathbf{x}) - D^\alpha u(\mathbf{y})|^p}{||\mathbf{x}-\mathbf{y}||_2^{n+ps}} d\mathbf{x} d\mathbf{y} \right)^{1/p}, \tag{2.14}$$

$$||u||_{W_p^{k+s}(\Omega)} := \left(||u||^p_{W_p^k(\Omega)} + |u|^p_{W_p^{k+s}(\Omega)} \right)^{1/p}. \tag{2.15}$$

Załóżmy, że $\Phi \in L_1(\mathbb{R}^n) \cap C(\mathbb{R}^n)$ spełnia

$$c_1(1 + ||\omega||_2^2)^{-\tau} \leqslant \hat{\Phi}(\omega) \leqslant c_2(1 + ||\omega||_2^2)^{-\tau}, \quad \omega \in \mathbb{R}^n \tag{2.16}$$

dla $\tau \in \mathbb{R}$ i $\tau > n/2$ oraz dwóch stałych dodatnich $c_1 < c_2$. Wówczas przestrzeń natywna $\mathcal{N}_\Phi(\mathbb{R}^n)$, odpowiadająca funkcji Φ, pokrywa się z przestrzenią Sobolewa $W_2^\tau(\Omega)$ i norma w przestrzeni natywnej, i norma w przestrzeni Sobolewa są równoważne.

Funkcjami Φ, które spełniają podany warunek, są ściśle dodatnio określone funkcje Wendlanda (zob. [30]) oraz np. funkcje typu *thin plate splines* $\Phi(r) = (-1)^{\beta+1}||r||^{2\beta} \log ||r||, \beta \in 2\mathbb{N}$, dla których

$$\hat{\Phi}(\omega) = 2^{n+2\beta-1}\pi^{n/2}\Gamma(n/2 + \beta)\beta!||\omega||^{-n-2\beta},$$

gdzie Γ oznacza funkcję Gamma Eulera (zob. [25]).

Przestrzeń natywna dla funkcji Gaussa

Dla funkcji Gaussa $\Phi(r) = e^{-\alpha||r||^2}$ mamy

$$\hat{\Phi}(\omega) = \left(\frac{\pi}{\alpha}\right)^{n/2} e^{-||\omega||^2/(4\alpha)}.$$

Wydaje się, że taka postać transformaty Fouriera funkcji Φ może być argumentem przeciw aproksymacji za pomocą funkcji Gaussa, gdyż każda funkcja z przestrzeni natywnej $\mathcal{N}_\Phi(\mathbb{R}^n)$ funkcji musi być zdominowana przez transformatę Fouriera $\hat{\Phi}$ w sensie (2.12). Warunek ten spełniają np. wszystkie funkcje o ograniczonym widmie, a przestrzeń tych funkcji odgrywa istotną rolę w teorii próbkowania, między innymi w twierdzeniu Shannona o próbkowaniu (zob. [32]).

2.2.4. Ograniczenia błędu dla dużej liczby punktów danych

Ograniczenia błędu, jakimi się zajmiemy w tym podpunkcie, dotyczą rzędu zbieżności procesu interpolacji. Oznacza to, że wyznaczymy funkcje, do jakich zbiega błąd funkcji interpolującej, gdy zbiór danych $\mathbf{X}$ staje się coraz gęstszy.

W prezentowanych ograniczeniach gęstość zbioru $\mathbf{X}$ w dziedzinie Ω mierzona jest globalnie za pomocą wskaźnika wypełnienia (ang. *fill distance*)

$$h_{\mathbf{X},\Omega} = \max_{\mathbf{x}\in\Omega} \min_{1\leqslant j\leqslant N} ||\mathbf{x} - \mathbf{x}_j||_2. \tag{2.17}$$

Określa on promień największej kuli, która może zostać umieszczona w dziedzinie Ω i nie zawierać żadnego punktu danych ze zbioru $\mathbf{X}$.

Wyniki tu przedstawione, dotyczące globalnej gęstości zbioru danych, przytaczamy wraz z istotnymi w ich dowodach lematami za [30] i [32].

Odtwarzanie wielomianów

Podstawowym narzędziem używanym w dowodach ograniczeń błędów interpolacji oraz aproksymacji jest odtwarzanie wielomianów. Odtwarzanie wielomianów przez proces interpolacyjny/aproksymacyjny definiuje się następująco:

Definicja 7
Proces, który określa dla każdego zbioru $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subseteq \Omega$ *rodzinę funkcji* $u_j = u_j^{\mathbf{X}} : \Omega \to \mathbb{R}$, $1 \leqslant j \leqslant N$ *zapewnia lokalne odtwarzanie wielomianów stopnia* l *na zbiorze* Ω, *jeśli istnieją stałe* $h_0, C_1, C_2 > 0$ *takie, dla których spełnione są warunki:*

1. $\sum_{j=1}^{N} p(\mathbf{x}_j)u_j = p$ *dla każdego* $p \in \pi_l(\mathbb{R}^n)|\Omega$,
2. $\sum_{j=1}^{N} |u_j(\mathbf{x})| \leqslant C_1$ *dla każdego* $\mathbf{x} \in \Omega$,
3. $u_j(\mathbf{x}) = 0$, *jeśli* $||\mathbf{x} - \mathbf{x}_j||_2 \geqslant C_2 h_{\mathbf{X},\Omega}$ *i* $\mathbf{x} \in \Omega$,

dla każdego $\mathbf{X}$, *dla którego* $h_{\mathbf{X},\Omega} \leqslant h_0$.

Jeśli proces gwarantuje lokalne odtwarzanie wielomianów to ograniczenie błędu aproksymacji funkcji tym procesem opisuje następujące twierdzenie.

Twierdzenie 5 (zob. [30])
Niech $\Omega \subset \mathbb{R}^n$ *jest ograniczony. Niech* Ω^* *będzie domknięciem zbioru* $\cup_{\mathbf{x}\in\Omega} B(x, C_2 h_0)$. *Zdefiniujmy*

$$s_{g,\mathbf{X}} = \sum_{j=1}^{N} g(x_j)u_j,$$

gdzie $\{u_j\}_{j=1}^{N}$ *jest lokalnym odtworzaniem wielomianów stopnia* m *na zbiorze* Ω. *Wówczas, jeśli* $g \in C^{m+1}(\Omega^*)$, *to istnieje stała* $c > 0$ *zależna tylko od stałych z definicji lokalnego odtwarzania wielomianów, taka że*

$$|g(\mathbf{x}) - s_{g,\mathbf{X}}(\mathbf{x})| \leqslant C h_{\mathbf{X},\Omega}^{m+1} |g|_{C^{m+1}(\Omega^*)} \tag{2.18}$$

dla każdego zbioru $\mathbf{X}$ *o gęstości* $h_{X,\Omega} < h_0$. *Seminorma po prawej stronie nierówności (2.18) jest określona następująco:*

$$|g|_{C^{m+1}(\Omega^*)} := \max_{|\alpha|=m+1} ||D^\alpha g||_{L_\infty(\Omega^*)}.$$

Na początku zadajmy sobie pytanie, jakie warunki muszą spełniać zbióry $\mathbf{X}$ i Ω, aby możliwe było lokalne odtwarzanie wielomianów stopnia nie większego niż m.

Warunkiem, jaki musi spełniać zbiór $\mathbf{X}$, jest unisolwentność względem przestrzeni $\pi_m(\mathbb{R}^n)$. Natomiast dziedzina Ω musi spełniać warunek stożka wewnętrznego.

Definicja 8 (warunek stożka wewnętrznego)
Mówimy, że zbiór $\Omega \subseteq \mathbb{R}^n$ spełnia warunek stożka, jeśli istnieje kąt $\theta \in (0, \pi/2)$ i promień $r > 0$ taki, że dla każdego $\mathbf{x} \in \Omega$ istnieje wektor jednostkowy $\xi(\mathbf{x})$ taki, że stożek

$$C(\mathbf{x}, \xi(\mathbf{x}), \theta, r) := \{\mathbf{x} + \eta\mathbf{y} : \mathbf{y} \in \mathbb{R}^n, ||\mathbf{y}||_2 = 1, \mathbf{y}^T\xi(\mathbf{x}) \geqslant \cos\theta, \eta \in [0, r]\}$$

zawiera się w Ω.

Twierdzenie 6 (o odtwarzaniu wielomianów)
Załóżmy, że $\Omega \subseteq \mathbb{R}^n$ jest zwarty i spełnia warunek stożka z promieniem $r > 0$ i kątem $\theta \in (0, \pi/2)$. Niech $m \in \mathbb{N}$ będzie ustalone. Załóżmy, że $h > 0$ i zbiór $\mathbf{X}$ spełniają następujące warunki:

1. $h \leqslant \frac{r \sin\theta}{4(1+\sin\theta)m^2}$, (2.19)
2. *dla każdej kuli $B(\mathbf{x}, h) \subseteq \Omega$ istnieje punkt $\mathbf{x}_j \in \mathbf{X} \cap B(\mathbf{x}, h)$.*

Wówczas dla każdego $\mathbf{x} \in \Omega$ istnieją liczby rzeczywiste $u_j(\mathbf{x})$ takie, że

$$p(\mathbf{x}) = \sum_{j=1}^{N} u_j(\mathbf{x}) p(\mathbf{x}_j) \tag{2.20}$$

dla każdego $p \in \pi_m(\mathbb{R}^n)$. Ponadto

$$\sum_{j=1}^{N} |u_j(\mathbf{x})| \leqslant 2. \tag{2.21}$$

Dowód tego twierdzenia, wykorzystujący pojęcie zbioru normującego oraz twierdzenie Hanha–Banacha o ograniczoności normy operatora liniowego, podany został w pracy [16]. Twierdzenie to określa stałe w definicji lokalnego odtwarzania wielomianów

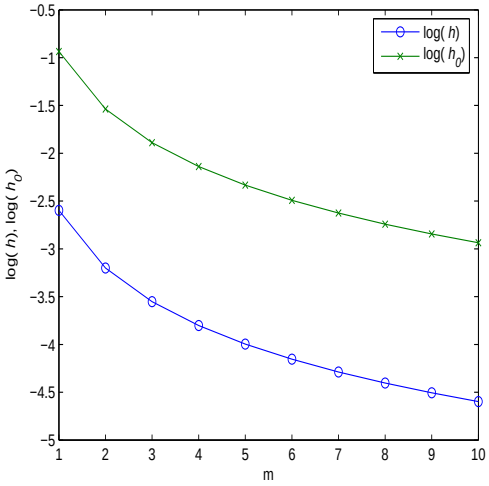
$$\begin{aligned} C_1 &= 2, \quad C_2 = \frac{16(1+\sin\theta)^2 m^2}{3\sin^2\theta}, \\ h_0 &= \frac{r}{C_2}. \end{aligned} \tag{2.22}$$

Zależność wielkości h i h_0 od stopnia wymaganego odtwarzania wielomianów jest pokazana na rysunku 2.1.

Funkcjonał potęgowy

Dowody wszystkich ograniczeń błędów interpolacji i aproksymacji, które wykorzystują funkcjonał potęgowy, wychodzą od ogólnej postaci błędu interpolacji

$$|g - s_{\mathbf{X},g}| \leqslant P_{\Phi,\mathbf{X}}(\mathbf{x})||g||_{\mathcal{N}_\Phi(\mathbb{R}^n)},$$



Rysunek 2.1. Zależność wskaźnika h w (2.19) oraz h_0 w (2.22) od maksymalnego stopnia odtwarzanych wielomianów w kuli jednostkowej

gdzie $P_{\Phi,\mathbf{X}}$ jest funkcjonałem potęgowym, który zdefiniujemy w tym podrozdziale. Funkcjonał potęgowy jest ściśle związany z reprezentacją Lagrange'a funkcji interpolującej dane ze zbioru $\mathbf{Z}$. Zacznijmy więc od twierdzenia o istnieniu reprezentacji Lagrange'a.

Twierdzenie 7
Załóżmy, że Φ jest warunkowo dodatnio określona względem przestrzeni $\pi_m(\mathbb{R}^n)$ na zbiorze $\Omega \subseteq \mathbb{R}^n$. Załóżmy, że $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ jest $\pi_m(\mathbb{R}^n)$-unisolwentny. Wówczas istnieją funkcje $u_j^* \in V_{\mathbf{X}}$ takie, że $u_j^*(\mathbf{x}_k) = \delta_{jk}$, gdzie

$$V_{\mathbf{X}} := \pi_m(\mathbb{R}^n) + \Big\{ \sum_{j=1}^{N} \alpha_j \Phi(\cdot, x_j) : \sum_{j=1}^{N} \alpha_j p(x_j) = 0, p \in \pi_m(\mathbb{R}^n) \Big\}.$$

Ponadto załóżmy, że istnieją funkcje v_j^*, $1 \leqslant j \leqslant Q$ takie, że wektory $u^*(x) = [u_1(\mathbf{x}), \ldots, u_N(\mathbf{x})]^T \in \mathbb{R}^N$ i $v^*(x) = [v_1(\mathbf{x}), \ldots, v_Q(\mathbf{x})]^T \in \mathbb{R}^Q$ tworzą rozwiązanie układu

$$\begin{pmatrix} A & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} u^*(\mathbf{x}) \\ v^*(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} R(\mathbf{x}) \\ S(\mathbf{x}) \end{pmatrix}, \tag{2.23}$$

gdzie

$$A \;=\; [\Phi(\mathbf{x}_i, \mathbf{x}_j)] \in \mathbb{R}^{N \times N},$$

$$\begin{aligned} P &= [p_i(\mathbf{x}_j)] \in \mathbb{R}^{N\times Q}, \\ R(\mathbf{x}) &= [\Phi(\mathbf{x},\mathbf{x}_1),\ldots,\Phi(\mathbf{x},\mathbf{x}_N))]^T \in \mathbb{R}^N, \\ S(\mathbf{x}) &= [p_1(\mathbf{x}),\ldots,p_Q(\mathbf{x}))]^T \in \mathbb{R}^Q. \end{aligned}$$

Na podstawie twierdzenia 7 funkcję interpolującą możemy zapisać w postaci

$$s_{g,\mathbf{X}} = \sum_{j=1}^{N} g(\mathbf{x}_j) u_j^*(\mathbf{x}). \tag{2.24}$$

Definicja 9
Załóżmy, że $\Omega \subseteq \mathbb{R}^n$ jest zbiorem otwartym, natomiast $\Phi \in C^{2k}(\Omega\times\Omega)$ jest warunkowo dodatnio określoną funkcją jądrową na zbiorze Ω względem $\pi_m(\mathbb{R}^n)$. Jeśli $\mathbf{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\} \subseteq \Omega$ jest $\pi_m(\mathbb{R}^n)$-unisolwentny, to dla każdego $\mathbf{x} \in \Omega$ funkcjonał potęgowy jest zdefiniowany jako

$$[P_{\Phi,\mathbf{X}}(\mathbf{x})]^2 := \Phi(\mathbf{x},\mathbf{x}) - 2\sum_{j=1}^{N} u_j^*(\mathbf{x})\Phi(\mathbf{x},\mathbf{x}_j) + \sum_{i,j=1}^{N} u_i^*(\mathbf{x}) u_j^*(\mathbf{x})\Phi(\mathbf{x}_i,\mathbf{x}_j)$$

Jeśli zdefiniujemy dla $\mathbf{x} \in \Omega$ formę kwadratową

$$\mathcal{Q}(u) := \Phi(\mathbf{x},\mathbf{x}) - 2\sum_{j=1}^{N} u_j\Phi(\mathbf{x},\mathbf{x}_j) + \sum_{i,j=1}^{N} u_i u_j \Phi(\mathbf{x}_i,\mathbf{x}_j),$$

to

$$[P_{\Phi,\mathbf{X}}(\mathbf{x})]^2 = \mathcal{Q}(u^*(\mathbf{x})) = \left| G(\cdot,\mathbf{x}) - \sum_{j=1}^{N} u_j^* G(\cdot,\mathbf{x}_j) \right|_{\mathcal{N}_\Phi(\Omega)}, \tag{2.25}$$

gdzie $G(\cdot,\mathbf{x})$ jest ściśle dodatnio określoną funkcja zdefiniowaną następująco

$$G(\cdot,\mathbf{x}) := \Phi(\cdot,\mathbf{x}) - \sum_{k=1}^{Q} p_k(\mathbf{x})\Phi(\cdot,\mathbf{x}_k).$$

Wyrażenie (2.25) jest więc normą funkcjonału obliczania wartości błędu interpolacji (2.3) w punkcie $\mathbf{x} \in \Omega$ w przestrzeni natywnej generowanej przez bazową funkcję radialną Φ.

Funkcja Lebesgue'a

Gdy mamy reprezentację Lagrange'a (2.24) lub odtwarzanie wielomianów zgodnie z twierdzeniem 6, wówczas funkcja Lebesgue'a zdefiniowana jest jako

$$\sum_{j=1}^{N} |u_j(\mathbf{x})|. \tag{2.26}$$

Funkcja ta jest istotna w analizie błędu interpolacji i aproksymacji funkcjami radialnymi, ponieważ za jej pomocą ogranicza się funkcjonał potęgowy. Dla interpolacji funkcjami Gaussa Schaback w [26] oraz Larsson i in. w [17] niezależnie dowiedli, że gdy szerokość funkcji Gaussa zdąża do nieskończoności, wówczas funkcja Lebesgue'a (2.26) zdąża do wielomianu.

Z twierdzenia 6 o odtwarzaniu wielomianów z przestrzeni $\pi_m(\mathbb{R}^n)$, wynika, że funkcja Lebesgue'a jest ograniczona:

$$\sum_{j=1}^{N} |u_j(\mathbf{x})| < C_1 = 2, \tag{2.27}$$

przy czym stała C_1 zależy od gęstości zbioru $\mathbf{X}$, tzn. od h_0. Natomiast h_0 zależy od m tak jak $o(1/m^2)$. Do dowodu zarówno tych zależności (zob. [16], [30]), jak i jednorodnego ograniczenia przez stałą 2, używa się nierówności Markowa o wielomianach i ich pochodnych, która zachodzi dla dowolnego wielomianu $p \in \pi_m(\mathbb{R}^n)$

$$|p'(t)| \leqslant m^2 ||p||_{L_\infty[-1,1]}, \quad t \in [-1,1]. \tag{2.28}$$

Z dowodu wiemy, jak dobrze zbiór $\mathbf{X}$ musi wypełniać dziedzinę Ω, aby zachowane było jednorodne ograniczenie $C_1 = 2$.

Nestety ograniczenia funkcji Lebesgue'a stałą 2 nie jest możliwe, jeśli chcemy, aby h_0 zależało od m tak jak $o(1/m)$. W takim wypadku ograniczenie funkcji Lebesgue'a przybiera formę zależności spektralnej. Mówi o tym następujący lemat i twierdzenie:

Lemat 1 (spektralna wersja nierówności Markowa)
Niech $\gamma_1 = 2$ oraz $\gamma_n = 2n(1+\gamma_{n-1})$ dla $n = 2, 3, \ldots$ Niech m i q będą liczbami naturalnymi, takimi, dla których zachodzi $q > \gamma_n(m+1)$. Niech Ω będzie sześcianem w $\mathbb{R}^n$. Niech dziedzina Ω będzie podzielona na q^n równych podsześcianów. Jeśli $\mathbf{X} \subseteq \Omega$ jest zbiorem $N \geqslant q^n$ punktów takich, dla których każdy podsześcian zawiera przynajmniej jeden z nich, to dla każdego $p \in \pi_m(\mathbb{R}^n)$ mamy

$$||p||_{L_\infty(\Omega)} \leqslant e^{2n\gamma_n(m+1)} ||p||_{L_\infty(\mathbf{X})}.$$

O odtwarzaniu wielomianów mamy następujące twierdzenie:

Twierdzenie 8
Niech $\Omega = W(\mathbf{x}_0, R)$ będzie sześcianem w $\mathbb{R}^n$. Istnieją stałe $c_0, c_2 > 0$ zależące tylko od Ω, dla których dla każdego $m \in \mathbb{N}$ i każdego $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subseteq \Omega$ z $h_{\mathbf{X},\Omega} \leqslant c_0/m$ możemy znaleźć funkcje $u_j : \Omega \to \mathbb{R}$ spełniające warunki:
1. $\sum_{j=1}^N u_j(\mathbf{x})p(\mathbf{x}_j) = p(\mathbf{x})$ *dla każdego* $\mathbf{x} \in \Omega$ *i* $p \in \pi_m(\mathbb{R}^n)$,
2. $\sum_{j=1}^N |u_j(\mathbf{x})| \leqslant e^{2n\gamma_n(m+1)}$ *dla każdego* $\mathbf{x} \in \Omega$,
3. $u_j(\mathbf{x}) = 0$ *jeśli* $||\mathbf{x} - \mathbf{x}_j||_2 > c_2 m h_{\mathbf{X},\Omega}$,

gdzie stała γ_n jest zdefiniowana jak w poprzednim lemacie.

Z dowodu tego twierdzenia mamy

$$h_0 = \frac{2R}{3q} = \frac{2R}{3\gamma_n(m+1)} =: \frac{c_0}{m}. \tag{2.29}$$

Jak wynika z punktu 3 twierdzenia 8, przeskalowaniu ulega również nośnik funkcji u_j w porównaniu z funkcjami u_j z twierdzenia 6. Jak widać z punktu 2 tego twierdzenia stała ograniczająca funkcję Lebesgue'a jest postaci $e^{2n\gamma_n(m+1)}$. Jest to wyrażenie bardzo szybko rosnące w zależnosci od m i od n.

Podane ograniczenia pokazują, jakiego rzędu wielkościami są funkcje Lebesgue'a. Dalej zobaczymy, że znajomość wartości tej funkcji w punkcie $\mathbf{x}$ wystarczy do oszacowania błędu interpolacji bez konieczności założeń o gęstości danych w otoczeniu punktu $\mathbf{x}$. W podrozdziale 2.3 natomiast pokażemy, jak wartość poszczególnych składowych Lagrange'a $u_j(\mathbf{x})$ może zostać użyta do oceny wpływu punktu $\mathbf{x}_j$ na jakość interpolacji na zbiorze $\mathbf{X}$.

Ograniczenia błędu wykorzystujące funkcjonał potęgowy

Ograniczenia błędu interpolacji i aproksymacji za pomocą radialnych funkcji bazowych wykorzystujące funkcjonał potęgowy wychodzą od następującego ogólnego twierdzenia.

Twierdzenie 9
Załóżmy, że $\Omega \subseteq \mathbb{R}^n$ jest zbiorem otwartym, natomiast $\Phi \in C^{2k}(\Omega \times \Omega)$ jest warunkowo dodatnio określoną funkcją jądrową na zbiorze Ω względem przestrzeni $\pi_m(\mathbb{R}^n)$. Zbiór $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subseteq \Omega$ jest $\pi_m(\mathbb{R}^n)$-unisolwentny. Niech $g \in \mathcal{N}_\Phi(\Omega)$ a jej interpolant oznaczmy przez $s_{g,\mathbf{X}}$. Wówczas dla każdego $\mathbf{x} \in \Omega$ błąd interpolacji funkcji g może być ograniczony przez

$$|g(\mathbf{x}) - s_{g,\mathbf{X}}(\mathbf{x})| \leqslant P_{\Phi,\mathbf{X}}(\mathbf{x})|g|_{\mathcal{N}_\Phi(\Omega)}. \tag{2.30}$$

Przytaczamy za [30] twierdzenie i pewne lematy dowodu najlepszego znanego, tzn. o najmniejszych znanych stałych, ograniczenia błędu interpolacji funkcjami Gaussa.

Twierdzenie 10
Funkcje kardynalne u_j^* obliczone dla $\mathbf{x} \in \Omega$ z układu (2.23) spełniają

$$P_{\Phi,\mathbf{X}}(\mathbf{x}) = \mathcal{Q}(u^*(\mathbf{x})) \leqslant \mathcal{Q}(u) \tag{2.31}$$

dla każdego $u \in \mathbb{R}^N$, gdzie mamy odtwarzanie wielomianów stopnia m, zgodnie z definicją 7, tzn.

$$\sum_{j=1}^{N} u_j p(x_j) = p(x) \qquad \text{dla każdego } p \in \pi_m(\mathbb{R}^n). \tag{2.32}$$

Mając zdefiniowane lokalne odtwarzanie wielomianów stopnia $l \geqslant m$, ograniczenie funkcjonału potęgowego można wyznaczyć, gdy zauważymy, że dla dowolnego wielomianu $p \in \pi_l(\mathbb{R}^n)$

$$\begin{aligned} p(0) &- 2\sum_{j=1}^{N} u_j(x)p(\mathbf{x}-\mathbf{x}_j) + \sum_{j=1}^{N}\sum_{k=1}^{N} u_j(\mathbf{x})u_k(\mathbf{x})p(\mathbf{x}_j-\mathbf{x}_k) \\ &= p(0) - 2p(\mathbf{x}-\mathbf{x}) + \sum_{j=1}^{N} u_j(\mathbf{x})p(\mathbf{x}_j-\mathbf{x}) \\ &= p(0) - 2p(0) + p(0) = 0. \end{aligned}$$

Korzystając z własności zerowania się funkcji kardynalnych z definicji 7, łatwo pokazać (zob. [32]), że

$$P_{\Phi,\mathbf{X}}^2 \leqslant F(u(\mathbf{x})) \leqslant (1+C_1)^2 ||\Phi - p||_{L_\infty(B(0,2C_2h_{\mathbf{X},\Omega}))}. \tag{2.33}$$

Dalej dla funkcji radialnej $\Phi = \phi(||\cdot||_2)$ wykorzystamy to, że każdy wielomian $p \in \pi_k(\mathbb{R})$ można zastąpić wielomianem $p(||\cdot||_2^2) \in \pi_{2k}(\mathbb{R}^n)$, uzyskując ograniczenie

$$P_{\Phi,\mathbf{X}}^2(\mathbf{x}) \leqslant \max_{0\leqslant t\leqslant 4C^2h^2} |\phi(\sqrt{t}) - p(t)|,$$

jeśli $h = h_{\mathbf{X},\Omega} \leqslant h_0(2k)$.

Dla zbioru Ω, będącego kulą o promieniu R, twierdzenie 6 o odtwarzaniu wielomianów redukuje się do następującego sformułowania:

Lemat 2
Załóżmy, że $\Omega = B(\mathbf{x}_0, R)$ *jest kulą o promieniu* $R > 0$. *Niech* $l \in \mathbb{N}$. *Dla ustalonego* C *takiego, że*

$$C \geqslant \frac{8(2+\sqrt{3})}{\sqrt{3}} l^2,$$

istnieje lokalne odtwarzanie wielomianów stopnia l *ze stałymi* $h_0 = R/C, C_1 = 2, C_2 = 2C$ *zgodnie z definicją 7.*

Ograniczenie błędu opisuje następne twierdzenie z wielkościami c_0 i c_2 takimi, dla których $h_0 = c_0/l^2$, $C_2 = c_2 l^2$:

Twierdzenie 11
Załóżmy, że na zbiorze Ω *mamy lokalne odtwarzanie wielomianów stopnia* l *dla każdego* $l \in \mathbb{N}$ *ze stałymi* $h_0 = c_0/l^2$, C_1 *niezależnego od* l, $C_2 = c_2 l^2$. *Niech* $\Phi(r) = e^{-\alpha||r||_2^2}$ z $\alpha > 0$. *Określmy* $\tilde{c}_0 := \min\{c_0, (32\alpha e c_2^2)^{-2/3}\}$. *Jeśli* $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subseteq \Omega$ *spełnia* $h_{\mathbf{X},\Omega} \leqslant \min\{1, \tilde{c}_0/4\}$, *to funkcjonał potęgowy może być ograniczony przez*

$$P^2_{\mathbf{X},\Omega}(\mathbf{x}) \leqslant \frac{(1+C_1)^2}{\sqrt{2\pi}} h^{\frac{\sqrt{\tilde{c}_0}}{4\sqrt{h}}}$$

dla każdego $\mathbf{x} \in \Omega$. *Tak więc dla* $g \in \mathcal{N}_\Phi(\Omega)$ *błąd interpolacji może być ograniczony wyrażeniem*

$$|g(\mathbf{x}) - s_{g,\mathbf{X}}(\mathbf{x})| \leqslant \frac{1+C_1}{(2\pi)^{1/4}} h^{\frac{\sqrt{\tilde{c}_0}}{8\sqrt{h}}} ||g||_\Phi.$$

Ograniczenia dla funkcji z przestrzeni Sobolewa

Inne podejście do konstrukcji ograniczenia błędu interpolacji lub aproksymacji za pomocą bazowych funkcji radialnych można zastosować dla funkcji z przestrzeni natywnej $\mathcal{N}_\Phi(\Omega)$ pokrywającej się z przestrzenią Sobolewa (zob. [20]). Twierdzenie o ograniczeniu błędu interpolacji funkcji z przestrzeni $W_2^\tau(\Omega)$ cytujemy za [30].

Twierdzenie 12
Załóżmy, że $\Omega \subseteq \mathbb{R}^n$ *jest ograniczony i ma brzeg spełniający warunek Lipschitza, jak również spełnia warunek stożka z promieniem* r *i kątem* θ. *Niech* $\mathbf{X} \subseteq \Omega$ *będzie zbiorem dyskretnym centrów a* $s_{g,\mathbf{X}}$ *będzie interpolantem. Załóżmy, że* Φ *jest funkcją jądrową przestrzeni Sobolewa, tzn. spełnia (2.16) z* $\tau = k + s$, *gdzie* k *jest dodatnia liczbą naturalną a* $0 < s \leqslant 1$. *Jeśli* $m \in \mathbb{N}_0$ *spełnia* $k > m + n/2$, *to błąd pomiędzy* $g \in W_2^\tau(\Omega)$ *a interpolantem* $s_{g,\mathbf{X}}$ *może być ograniczony w następujący sposób:*

$$|g - s_{g,\mathbf{X}}|_{W_q^m} \leqslant C h_{\mathbf{X},\Omega}^{\tau - m - n(1/2 - 1/q)_+} ||g||_{W_2^\tau(\Omega)}$$

dla dostatecznie gęstego zbioru $\mathbf{X}$.

W dowodzie tego twierdzenia nie wykorzystuje się funkcjonału potęgowego. Korzysta się natomiast:

1. z postaci normy, która zawiera operatory różniczkowania D^k dla $k \leqslant |\alpha|$ określone tak jak w (2.13),
2. z twierdzenia o odtwarzaniu pochodnych wielomianów, tzn. z uogólnienia twierdzenia o odtwarzaniu wielomianów,
3. dla funkcji

$$u(\mathbf{x}) = g(\mathbf{x}) - s_{g,\mathbf{X}}(\mathbf{x}) \tag{2.34}$$

z aproksymacji uśrednionym wielomianem Taylora

$$Q_k u(\mathbf{x}) := \sum_{|\alpha|<k} \frac{1}{\alpha!} \int_{B(0,\rho)} D^\alpha u(\mathbf{y})(\mathbf{x}-\mathbf{y})^\alpha \phi_\rho(\mathbf{y}) d\mathbf{y}, \tag{2.35}$$

gdzie $\phi_\rho \in C_0^\infty(\mathbb{R}^n)$ ma nośnik $B(0,\rho)$ i w sensie całkowym przybliża jedynkę.

Korzystając z podanych własności dowodzi się, że dla $u \in W\ {}_p^{k+s}(\Omega)$, która znika na punktach ze zbioru $\mathbf{X}$ dla $k > n/p - m$, zachodzi kluczowe ograniczenie

$$|u|_{W_q^m(\Omega)} \leqslant c h^{k+s-|\alpha|-n(1/p-1/q)_+} |u|_{W_p^{k+s}(\Omega)}, \tag{2.36}$$

gdzie h jest miarą gęstości siatki (2.17). Dowód polega na podzieleniu dziedziny Ω na zbiory gwiaździsto-kształtne $\mathcal{D}$ o średnicy $\mathcal{O}(h_{X,\Omega})$. Na każdym obszarze $\mathcal{D}$ korzysta się z aproksymacji uśrednionym wielomianem Taylora.

Definicja 10
Zbiór $\mathcal{D}$ *nazywamy gwiaździsto-kształtnym względem kuli* $B(\mathbf{x}_c,\rho) := \{\mathbf{x} \in \mathbb{R}^n : ||\mathbf{x}-\mathbf{x}_c|| \leqslant \rho\}$, *jeśli dla każdego* $\mathbf{x} \in \mathcal{D}$ *domknięta otoczka wypukła zbioru* $\{\mathbf{x}\} \cup B$ *jest zawarta w* $\mathcal{D}$. *Jeśli* $\mathcal{D}$ *jest ograniczony, to jego parametr kawałkowatości* γ_c *(ang. chunkiness parameter) jest zdefiniowany jako stosunek średnicy* $d_\mathcal{D}$ *do promienia* $\rho_{\max}$ *największej kuli, względem której zbiór* $\mathcal{D}$ *jest gwiaździsto-kształtny.*

Zbiór gwiaździsto-kształtny spełnia warunek stożka, o czym mówi następujący lemat:

Lemat 3
Jeśli $\mathcal{D}$ *jest ograniczony, gwiaździsto-kształtny względem kuli* $B(\mathbf{x}_c,\rho)$ *i zawarty w kuli* $B(\mathbf{x}_c,R)$, *to spełnia warunek stożka z promieniem* ρ *i kątem* $\vartheta = 2\arcsin[\rho/(2R)]$.

Obszarami gwiaździsto-kształtnymi pokrywamy dziedzinę Ω. Na każdym takim obszarze aproksymujemy funkcje u za pomocą wielomianu (2.35). Dowodzi się wówczas następującego ograniczenia błędu takiej aproksymacji na każdym obszarze $\mathcal{D}$:

Lemat 4
Niech $0 < s \leqslant 1$ i $m \in \mathbb{N}$. Niech $1 < p < \infty$ i $k > m + n/p$ lub $p = 1$ i $k \geqslant m + n$. Dla $u \in W_p^{k+s}(\mathcal{D})$ mamy

$$||u - Q_{k+1}u||_{W_\infty^m(\mathcal{D})} \leqslant C(1+\gamma_c)^{n(1+1/p)} d_{\mathcal{D}}^{k+s-m-n/p} |u|_{W_p^{k+s}(\mathcal{D})},$$

ze stałą $C > 0$ zależną tylko od k, n i p.

W dalszej części dowodu korzysta się z tego, że funkcje u znikają na zbiorze $\mathbf{X}$. Korzysta się przy tym z własności odtwarzania pochodnych wielomianu (uogólnienie twierdzenia 6). Mamy więc następujące lematy:

Lemat 5
Niech k będzie stałą dodatnią, $1 \leqslant p < \infty, 0 < s \leqslant 1, 1 \leqslant q \leqslant \infty$, i niech $m \in \mathbb{N}_0$ spełniają $1 < p < \infty$ i $k > m + n/p$ lub $p = 1$ i $k \geqslant m + n$. Niech również $\mathbf{X} \subseteq \mathcal{D}$ będzie zbiorem dyskretnym spełniającym warunek 1. i 2. odtwarzania wielomianów z twierdzenia 6 z $h > 0$. Jeśli $u \in W_p^{k+s}(\mathcal{D})$ spełnia $u|\mathbf{X}$, to

$$|u|_{W_q^m(\mathcal{D})} \leqslant C d_{\mathcal{D}}^{k+s-m+n(1/q-1/p)} |u|_{W_p^{k+s}(\mathcal{D})},$$

przy czym stała C zależy tylko od k, n, p, m i kąta ϑ z warunku stożka dla $\mathcal{D}$.

Przejście do jednorodnego ograniczenia w całej dziedzinie Ω wymaga określenia parametrów pokrycia zbioru Ω obszarami $\mathcal{D}$.

Lemat 6
Wprowadźmy wielkości

$$\begin{aligned}
\vartheta &:= 2 \arcsin\left(\frac{\sin\theta}{4(1+\sin\theta)}\right), \\
Q(k,\theta) &:= \frac{\sin\theta \sin\vartheta}{8k^2(1+\sin\theta)(1+\sin\vartheta)}, \\
R &:= Q(k,\theta)^{-1}, \\
\rho &:= \frac{\sin\theta}{2(1+\sin\theta)} R.
\end{aligned}$$

Ponadto zdefiniujmy zbiory

$$T_\rho := \{\mathbf{t} \in (2\rho/\sqrt{n})\mathbb{Z}^n : B(\mathbf{t}, \rho) \subseteq \Omega\}$$

oraz

$$\mathcal{D}_t = \{\mathbf{x} \in \Omega : \mathrm{co}(\{\mathbf{x}\} \cup B(\mathbf{t}, \rho)) \in \Omega \cap B(\mathbf{t}, R)\}, \qquad dla\ \mathbf{t} \in T_\rho,$$

gdzie $\mathrm{co}(A)$ *jest otoczką wypukła zbioru* A.
Załóżmy, że $h = h_{\mathbf{X},\Omega}$ *spełnia* $h \leqslant Q(k,\theta)r$. *Następujące stwierdzenia są prawdziwe:*

1. *każdy zbiór* $\mathcal{D}_\mathbf{t}$ *jest gwiaździsto-kształtny względem kuli* $B(\mathbf{t}, \rho) \subseteq \mathcal{D}_\mathbf{t} \subseteq \Omega \cap B(\mathbf{t}, R)$,
2. *każdy zbiór* $\mathcal{D}_\mathbf{t}$ *spełnia warunek stożka z kątem* ϑ *z promieniem* ρ,
3. $\Omega = \bigcup_{\mathbf{t}\in T_\rho} \mathcal{D}_\mathbf{t}$ *i* $d_{\mathcal{D}_t} \leqslant 2R = 2h/Q(k,\theta)$,
4. $\sum_{\mathbf{t}\in T_\rho} \chi_{\mathcal{D}_\mathbf{t}} \leqslant M_1$,
5. $|T_\rho| \leqslant M_2\rho^{-n}$,

gdzie χ_B *oznacza funkcję charakterystyczną zbioru* B *a stałe* M_1, M_2 *zależą tylko od* k, θ *i* n.

Korzystając z tej geometrycznej konstrukcji pokrycia, zbiory Ω dla $\mathbf{X} \subseteq \Omega$ z gęstością siatki h spełniającą $h < Q(k,\theta)r$, w rezultacie otrzymujemy ograniczenie (2.36) na całym zbiorze Ω.

Korzystając z (2.36) dla $h < Q(k,\theta)r$ i $p = q = 2$, otrzymujemy wynik z twierdzenia 12.

Aproksymacja

Analogiczny mechanizm dowodzenia jak dla interpolacji można zastosować do wyznaczenia ograniczenia błędu zregularyzowanej aproksymacji średniokwadratowej, tzn. rozwiązania zagadnienia (2.5) z $\Phi \in W_p^\tau$ i λ takiego, dla którego $|g(\mathbf{x}_j) - s_{g,\mathbf{X}}(\mathbf{x}_j)| \leqslant \epsilon$ dla $1 \leqslant j \leqslant N$. Wówczas mamy ograniczenie

$$||g - s_{g,\mathbf{X}}||_{L_\infty(\Omega)} \leqslant Ch_{\mathbf{X},\Omega}^{\tau - n/p}|g|_{W_p^\tau(\Omega)} + 2\epsilon.$$

Wynika to z twierdzenia analogicznie dowodzonego jak twierdzenie 12.

Twierdzenie 13

Załóżmy, że Ω *jest ograniczony i spełnia warunek stożka. Niech* k *będzie stałą dodatnią,* $0 < s \leqslant 1, 1 \leqslant p < \infty, 1 \leqslant q \leqslant \infty$ *i niech* $m \in \mathbb{N}_0$ *spełnia* $k > m + n/p$ *dla* $p > 1$ *lub dla* $p = 1, k \geqslant m + n$. *Ponadto, niech* $\mathbf{X} \subseteq \Omega$ *będzie zbiorem*

dyskretnym z gęstością siatki h *spełniającą* $h < Q(k,\theta)r$. *Jeśli* $u \in W_p^{k+s}(\Omega)$ *spełnia* $|u|_{\mathbf{X}} \leqslant \epsilon$, *to mamy ograniczenie*

$$|u|_{W_q^m(\Omega)} = C\left(h^{k+s-m-n(1/p-1/q)_+}|u|_{W_p^{k+s}(\Omega)} + h^{-m}||u|\mathbf{X}||_\infty\right), \tag{2.37}$$

gdzie $(x)_+ = \max(x,0)$. *Stała* C *zależy tylko od* k,n,p,q,m *i* θ.

Rozważając więc zagadnienie (2.5), możemy sformułować następujące twierdzenie:

Twierdzenie 14 (zob. [31])
Załóżmy, że $\mathcal{H} \subseteq C(\Omega)$ *jest unormowaną przestrzenią liniową funkcji ciągłych. Załóżmy ponadto, że* $L : \mathcal{H} \to \mathcal{G}$ *jest przekształceniem liniowym w przestrzeń unormowaną* $\mathcal{G}$. *Niech* $\mathbf{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\} \subseteq \Omega$ *i* $\{y_1,\ldots,y_N\} \in \mathbb{R}$ *definiują funkcjonały*

$$E(s) := \sum_{j=1}^{N}[y_j - s(\mathbf{x}_j)]^2, \quad J(s) := ||Ls||_{\mathcal{G}}^2, \quad s \in \mathcal{H}.$$

Załóżmy, że $s_\lambda \in \mathcal{H}$ *jest rozwiązaniem zagadnienia*

$$\min_{s\in\mathcal{H}} E(s) + \lambda J(s)$$

dla ustalonego $\lambda > 0$. *Załóżmy ponadto, że istnieje funkcja* $s_0 \in \mathcal{H}$ *z* $E(s_0) = 0$ *i* $J(s_0) \leqslant J(g)$. *Przy danych założeniach prawdziwe są ograniczenia*

$$\begin{aligned} J(s_\lambda) &\leqslant J(g) \\ |g(x_j) - s_\lambda(x_j)| &\leqslant \sqrt{\lambda J(g)}, \quad 1 \leqslant j \leqslant N. \end{aligned}$$

2.2.5. Ograniczenia błędu bez założeń o gęstości siatki

Ograniczenia błędu oparte na ogólnej postaci (2.30) bez założeń o gęstości zbioru danych rozważane są w [11]. Opierają się one na pojęciu funkcji wzrostu wielomianów zdefiniowanej następująco:

Definicja 11
Dla zbioru $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subseteq \Omega$ *i* $\mathbf{Y} \subseteq \mathbf{X}$ *definiujemy funkcję wzrostu dla przestrzeni wielomianów* $\pi_q(\mathbb{R}^n)$ *na zbiorze* $\mathbf{Y}$ *w punkcie* $\mathbf{x}$ *jako*

$$\rho_q(\mathbf{x},\mathbf{Y}) = \max\{|p(\mathbf{x})| : p \in \pi_q(\mathbb{R}^n), ||p|_{\mathbf{Y}}||_\infty \leqslant 1\}.$$

Funkcja wzrostu $\rho_q(\mathbf{x},\mathbf{Y})$ ma skończoną wartość dla każdego $\mathbf{x}$, jeśli $\mathbf{Y}$ jest $\pi_q(\mathbb{R}^n)$-unisolwentny. W przeciwnym wypadku $\rho_q(\mathbf{x},\mathbf{Y}) = \infty$.

Twierdzenie 15
Załóżmy, że dany mamy zbiór $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subseteq \Omega$ *oraz* $y_i = g(\mathbf{x}_i)$, $i = 1, \ldots, N$ *dla funkcji* $g \in \mathcal{N}_\Phi(\Omega)$. *Dla rozwiązania zadania interpolacyjnego (2.3) i dowolnego niepustego podzbioru* $\mathbf{Y} \subseteq \mathbf{X}$ *oraz dowolnego* $q \geqslant \max\{m, 0\}$ *mamy*

$$|g(x) - s_{g,\mathbf{X}}| \leqslant (1 + \rho_q(\mathbf{x}, \mathbf{Y}))\sqrt{E(\Phi, \pi_q^n)_{C(B(\mathbf{0}, \operatorname{diam}(\mathbf{Y} \cup \{\mathbf{x}\}))}}||g||_{\mathcal{N}_\Phi(\Omega)}, \quad \mathbf{x} \in \mathbb{R}^n \tag{2.38}$$

gdzie:

1. $\rho_q(\mathbf{x}, \mathbf{Y})$ *jest funkcją wzrostu wielomianów dla przestrzeni* $\pi_q(\operatorname{co}(\mathbf{Y}))$,
2. $B(\mathbf{0}, r)$ *oznacza kulę w* $\mathbb{R}^n$ *o środku w punkcie* $\mathbf{0}$ *i promieniu* r,
3. $E(\Phi, \pi_q^n)_{C(B_{\mathbf{x},\mathbf{Y}})}$ *zdefiniowane jest jako*

$$E(F, \mathcal{S})_{C(G)} := \inf_{g \in \mathcal{S}} ||F - g||_{C(G)}$$

i oznacza błąd aproksymacji jednostajnej funkcji F *za pomocą funkcji z* $\mathcal{S}$ *określonej na* $G \subset \mathbb{R}^n$. *W ograniczeniu (2.38) mamy* $G = B(\mathbf{0}, \operatorname{co}(\mathbf{Y} \cup \{\mathbf{x}\}))$, $\mathcal{S} = \pi_q^n(G)$ *oraz* $F = \Phi$.

Zauważmy, że funkcja $\rho_q(\mathbf{x}, \mathbf{Y})$ jest równa funkcji Lebesgue'a (2.26), gdy $|\mathbf{Y}| = \dim(\pi_q(\mathbb{R}^n))$. Oszacowanie wartości $\rho_q(\mathbf{x}, \mathbf{X})$ dla dowolnego $\mathbf{x} \in \Omega$ znaleźć można w [12].

2.2.6. Wskaźniki gęstości danych

W rozdziale tym omówimy dwa wskaźniki mierzące lokalną jakość wypełnienia dziedziny Ω przez punkty zbioru danych $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$.

Norma siatki

Norma siatki (ang. *mesh norm*) jest wielkością określającą jak dobrze zbiór danych $\mathbf{X} \subseteq \Omega$ wypełnia dziedzinę Ω. Jest definiowana w dwóch wersjach:

1. lokalnej [24] – dla parametru $\rho > 0$ dla każdego punktu $\mathbf{x} \in \Omega$

$$h_{\rho,\mathbf{X},\Omega}(\mathbf{x}) := \max_{\mathbf{y} \in B(\mathbf{x},\rho) \cap \Omega} \min_{\mathbf{x}_i \in \mathbf{X}} ||\mathbf{y} - \mathbf{x}_i||_2, \tag{2.39}$$

   gdzie $B(\mathbf{x}, \rho)$ oznacza kulę o środku $\mathbf{x}$ i promieniu ρ,
2. globalnej [30]

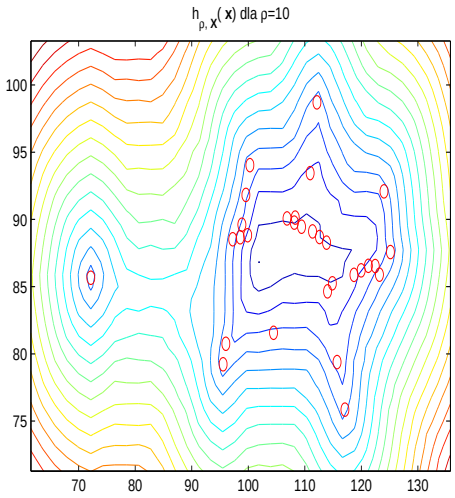
$$h_{\mathbf{X},\Omega} := \max_{\mathbf{y} \in \Omega} \min_{\mathbf{x}_j \in \mathbf{X}} ||\mathbf{y} - \mathbf{x}_j||_2. \tag{2.40}$$

W wersji lokalnej wskaźnik ten określa promień największej kuli niezawierającej punktów danych ze zbioru $\mathbf{X}$ i zawartej w kuli o środku $\mathbf{x}$ i promieniu ρ. W wersji

globalnej określa promień największej kuli niezawierającej danych ze zbioru $\mathbf{X}$, zawartej w dziedzinie Ω.

Mając dany zbiór $\mathbf{X}$ oraz promień ρ, do obliczenia lokalnej normy siatki można wykorzystać algorytm triangulacji Delaunaya [13]. Niestety, konstrukcja triangulacji Delaunaya jest możliwa w przestrzeni $\mathbb{R}^n$ tylko wtedy, gdy zbiór danych złożony jest z punktów w pozycji ogólnej, tzn. nie zawiera $n+1$ punktów współliniowych ani $n+2$ punktów położonych na okręgu. Ten warunek nie będzie zazwyczaj spełniony, gdy zbiór danych pochodzi ze ścieżki algorytmu optymalizacyjnego.

Rysunek 2.2 pokazuje przykładowy wykres wskaźnika $h_{\rho,\mathbf{X},\Omega}$ dla 30 punktów przykładowego zbioru danych dwuwymiarowych.



Rysunek 2.2. $h_{\rho,\mathbf{X},\Omega}(\mathbf{x})$ dla $\rho = 10$ dla przykładowego zbioru 30 punktów danych z dwuwymiarowego procesu optymalizacyjnego

Wskaźnik otoczenia $\gamma_{\mathbf{X}}(\mathbf{x})$

Drugim rozważanym przez nas wskaźnikiem mierzącym lokalną jakość rozkładu punktów zbioru $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ w otoczeniu punktu $\mathbf{x}$ jest wskaźnik mierzący długości odcinków $\mathbf{x}\bar{\mathbf{x}}\mathbf{x}_i \quad (i = 1, \ldots, N)$ i kąty $\angle\mathbf{x}_i\mathbf{x}\mathbf{x}_j \quad (i, j = 1, \ldots, N;\ i < j)$ między punktami zbioru $\mathbf{X}$. Lokalność takiego wskaźnika może być uzyskana w ten sposób, że nadaje się większą wagę krótszym odcinkom $\mathbf{x}\bar{\mathbf{x}}\mathbf{x}_i$ oraz większym kątom $\angle\mathbf{x}_i\mathbf{x}\mathbf{x}_j$. Wskaźnik o takich własnościach może być zdefiniowany następująco (zob. [4]):

$$\gamma_{\mathbf{X}}(\mathbf{x}) = \sum_{\substack{i,j \\ j<i}}^{N} a_{ij} W_{ij} \Big/ \sum_{\substack{i,j \\ j<i}}^{N} W_{ij}, \tag{2.41}$$

gdzie

$$a_{ij} = \frac{d_{ij}}{r_i + r_j}, \qquad \text{dla} \qquad d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||_2, \quad \text{i} \quad r_i = ||\mathbf{x} - \mathbf{x}_i||_2,$$

oraz wagi W_{ij} są zdefiniowane jako

$$W_{ij} = \frac{1}{r_i + r_j}.$$

W przeciwieństwie do wskaźników zdefiniowanych w podrozdziale 2.2.6, które zależą od gęstości wypełnienia kuli $B(\mathbf{x}, \rho)$ (w wypadku $h_{\rho,\mathbf{X},\Omega}$) oraz gęstości wypełnienia całej dziedziny Ω przez punkty zbioru $\mathbf{X}$ (w wypadku $h_{\mathbf{X},\Omega}$), wskaźnik $\gamma_{\mathbf{X}}(\mathbf{x})$ mierzy, jak punkty danych otaczają punkt $\mathbf{x}$. Wartość $\gamma_{\mathbf{X}}(\mathbf{x})$ jest tym większa, im więcej punktów znajduje się w bliskim otoczeniu punktu $\mathbf{x}$ i są ułożone tak, że występuje dużo kątów rozwartych $\angle \mathbf{x}_i, \mathbf{x}, \mathbf{x}_j$ $(i, j = 1, \ldots, N;$ $i < j)$ w bliskim otoczeniu punktu $\mathbf{x}$.

Własności takie wskaźnik $\gamma_{\mathbf{X}}(\mathbf{x})$ zawdzięcza własnościom funkcji a_{ij} i W_{ij}. Funkcje a_{ij} przyjmują największą wartość równą 1 dla punktów $\mathbf{x}$ położonych na odcinku $\overline{\mathbf{x_i x_j}}$. Maksimum jest więc osiągane, gdy kąt $\angle \mathbf{x}_i, \mathbf{x}, \mathbf{x}_j = \pi$. Wzmocnienie wpływu par punktów bliżej położonych punktu $\mathbf{x}$ jest uzyskiwane przez funkcję wagową W_{ij}, która maksymalną wartość równą $1/d_{ij}$ przyjmuje również na odcinku $\overline{\mathbf{x_i x_j}}$. Im bliżej więc odcinka $\overline{\mathbf{x_i x_j}}$ i im mniejsze d_{ij}, tym większy wpływ pary punktów $\mathbf{x}_i, \mathbf{x_j}$ na wartość wskaźnika dla punktu $\mathbf{x}$.

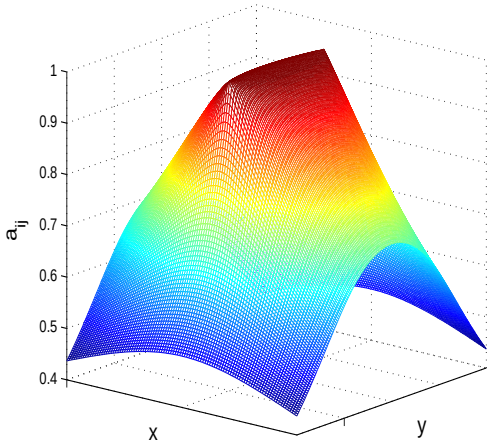
Na rysunku 2.3 pokazano wykres funkcji a_{ij} oraz W_{ij}, natomiast na rysunku 2.4 przedstawiono wykres wskaźnika $\gamma_{\mathbf{X}}(\mathbf{x})$ dla 30 punktów przykładowego zbioru danych z procesu optymalizacyjnego dwóch zmiennych.

2.2.7. Regularyzacja Tichonowa

Zagadnienie treningu sieci z radialnymi funkcjami bazowymi jest zagadnieniem źle postawionym w sensie Hadamarda [14]. Zadanie jest zadaniem źle postawionym (ang. *ill-posed*), jeśli nie jest zadaniem dobrze postawionym, czyli gdy nie jest spełniony co najmniej jeden z warunków następującej definicji.

Definicja 12
Zadanie nazywamy dobrze postawionym, jeśli zachodzą następujące warunki:

1. *rozwiązanie zadania istnieje,*



Rysunek 2.3. Wykres funkcji $a_{ij}(\mathbf{x})$. Maksymalna wartość przyjmowana jest na odcinku $\overline{\mathbf{x}_i\mathbf{x}_j}$. Wykres funkcji wagowej W_{ij} jest analogiczny. Maksymalna wartość funkcji wagowej również jest przyjmowana na odcinku $\overline{\mathbf{x}_i\mathbf{x}_j}$ i wynosi $1/d_{ij}$

2. *rozwiązanie zadania jest jednoznaczne,*
3. *rozwiązanie zadania zależy w sposób ciągły od danych.*

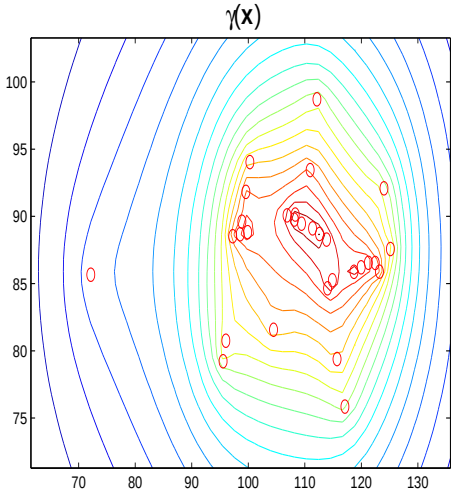
Zadanie treningu sieci z radialnymi funkcjami bazowymi przez rozwiązanie zagadnienia interpolacyjnego nie spełnia zazwyczaj warunku 3. Wynika to z tego, że macierz interpolacyjna A jest macierzą bardzo źle uwarunkowaną. Złe uwarunkowanie jest większe dla funkcji dodatnio określonych i nieskończenie wiele razy różniczkowalnych, tzn. np. dla funkcji Gaussa czy odwrotnej funkcji multikwadratowej.

Dla sieci trenowanych przez rozwiązanie zagadnienia aproksymacyjnego nie jest spełniony ponadto warunek 2.

Rozważmy metodę regularyzacji Tichonowa z operatorem ograniczeń $L \in \mathbb{R}^{r\times N}$. Polega ona na rozwiązaniu zagadnienia minimalizacyjnego

$$\min_{\mathbf{w}\in\mathbb{R}^N}\{||A\mathbf{w}-\mathbf{y}||^2+\lambda^2||L\mathbf{w}||_2^2\}, \tag{2.42}$$

gdzie r jest rzędem macierzy A a $\lambda \in \mathbb{R}$ jest nazywane parametrem regularyzacji.



Rysunek 2.4. Wykres wskaźnika $\gamma_{\mathbf{X}}(\mathbf{x})$ dla przykładowego zbioru 30 punktów danych z dwuwymiarowego procesu optymalizacyjnego

Rozwiązanie tego problemu można zapisać za pomocą uogólnionego rozkładu SVD (GSVD) macierzy A i L. Rozkład GSVD dla macierzy A i L definiujemy (zob. [14]) jako

$$A = U \begin{pmatrix} \Sigma & 0 \\ 0 & I_{N-r} \end{pmatrix} X^{-1}, \qquad L = V X^{-1}.$$

Kolumny macierzy $U \in \mathbb{R}^{N\times N}$ oraz $U \in \mathbb{R}^{r\times r}$ są ortonormalne, natomiast kolumny macierzy $X \in \mathbb{R}^{N\times N}$ są ortogonalne względem macierzy $A^T A$, tzn.

$$X^T A^T A X = \begin{pmatrix} \Sigma^2 & 0 \\ 0 & I_{N-r} \end{pmatrix}$$

oraz spełniają

$$X^T L^T L X = \begin{pmatrix} M^2 & 0 \\ 0 & 0 \end{pmatrix},$$

gdzie macierze $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$ oraz $M = \operatorname{diag}(\mu_1, \ldots, \mu_r)$ mają na przekątnej nieujemne elementy uporządkowane tak, że zachodzi

$$0 \leqslant \sigma_1 \leqslant \cdots \leqslant \sigma_r \leqslant 1, \qquad 1 \geqslant \mu_1 \geqslant \cdots \geqslant \mu_r \geqslant 0$$

przy dodatkowym warunku normalizacji

$$\sigma_i^2 + \mu_i^2 = 1, \quad i = 1, \ldots, r.$$

Wówczas

$$\gamma_i = \sigma_i/\mu_i \quad i = 1, \ldots, r$$

nazywamy uogólnionymi wartościami szczególnymi. Jeśli macierz A jest pełnego rzędu, to rozwiązanie $\mathbf{w}_\lambda$ podanego problemu zapisać można za pomocą rozkładu SVD

$$\mathbf{w}_\lambda = XF \begin{pmatrix} \Sigma^{-1} & 0 \\ 0 & I_{N-r} \end{pmatrix} U^T \mathbf{y},$$

gdzie $F = \mathrm{diag}(f_i)$. Elementy diagonalne f_i nazywają się współczynnikami filtrującymi.

Rozwiązanie zregularyzowane $\mathbf{w}_\lambda$ oraz odpowiadający mu wektor residuów $\mathbf{y} - A\mathbf{w}_\lambda$ można zapisać więc jako

$$\mathbf{w}_\lambda = \sum_{i=1}^{r} f_i \frac{\mathbf{u}_i^T \mathbf{y}}{\sigma_i} x_i + \sum_{k=r+1}^{N} \mathbf{u}_i^T \mathbf{y} x_i \tag{2.43}$$

i

$$L\mathbf{w}_\lambda = \sum_{i=1}^{r} f_i \frac{\mathbf{u}_i^T \mathbf{y}}{\gamma_i} \mathbf{v}_i$$

oraz

$$\mathbf{y} - A\mathbf{w}_\lambda = \sum_{i=1}^{r} (1 - f_i) \mathbf{u}_i^T \mathbf{y} \mathbf{u}_i + (I_N - UU^T)\mathbf{y}.$$

Współczynniki filtrujące mają postać

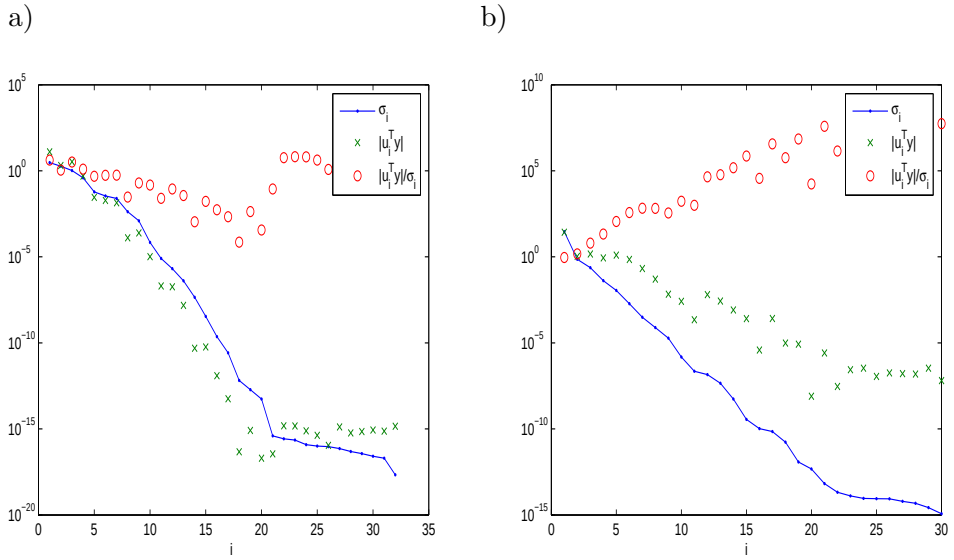
$$f_i = \frac{\gamma_i^2}{\gamma_i^2 + \lambda^2}, \quad i = 1, \ldots, r.$$

Redukują one w rozwiązaniu $\mathbf{w}_\lambda$ wpływ składowych odpowiadających wartościom szczególnym $\sigma_i \leqslant \sigma_p$ takim, dla których $\sigma_p < \lambda < \sigma_{p-1}$, ponieważ $f_i/\sigma_i \ll 1/\sigma_i$ dla każdego $p \leqslant i \leqslant N$. Tak więc składowe odpowiadające małym wartościom szczególnym, czyli te, które wzmacniają zaburzenie danych, są odfiltrowywane.

Metoda regularyzacji Tichonowa jest metodą rozwiązywania zadań źle postawionych i została zaproponowana jako metoda rozwiązywania zagadnień odwrotnych sformułowanych w postaci całki Fredholma

$$g(s) = \int_a^b K(s,t) g(t) dt.$$

Szukanym rozwiązaniem zagadnienia odwrotnego jest funkcja $g(t)$ dla danej funkcji $g(s)$ i funkcji jądrowej $K(s,t)$. Dyskretyzacja równania Fredholma daje zawsze macierz źle uwarunkowaną.



Rysunek 2.5. Warunek Picarda dla równania całkowego Fredholma [14] (a). Warunek Picarda dla aproksymacji funkcjami Gaussa i zbioru 30 punktów danych (b)

Warunek Picarda

Warunkiem wystarczającym, aby metoda regularyzacji Tichonowa dawała interpretowalne rozwiązanie dla zagadnienia odwrotnego zadanego całką Fredholma jest spełnienie dyskretnego warunku Picarda. Według dyskretnego warunku Picarda ciąg wielkości $|\mathbf{u}_i^T\mathbf{y}|$, $i = 1, 2, \ldots, r$ musi maleć szybciej niż ciąg σ_i, $i = 1, \ldots, N$. Jest to warunek, który łączy macierz równania z wektorem prawej strony. Dzięki temu, że $\mathbf{u}_i^T\mathbf{y} < \sigma_i$ dla wszystkich $i > p$ dla pewnego p współczynniki filtrujące zmniejszają w rozwiązaniu wkład wektorów własnych odpowiadających małym wartościom własnym dla $i > p$. Dla zadania aproksymacji funkcjami radialnymi warunek ten nie jest spełniony (zob. rys. 2.5). Dla zagadnienia aproksymacyjnego rzeczywiste zmniejszenie wkładu wektorów własnych zachodzi dla $\sigma_p < \lambda < \sigma_{p-1}$ i wektorów własnych o indeksach $i > p + k$, gdzie k jest najmniejszą liczbą naturalną, dla której $u_{p+k}^T y > \lambda$.

Metoda uogólnionej walidacji krzyżowej

Najpopularniejszą metodą wyboru parametru regularyzacji λ jest metoda uogólnionej walidacji krzyżowej (ang. *Generalized Cross Validation (GCV)* – zob. [29]). Jest to metoda, która nie wymaga znajomości *a priori* ani zaburzenia danych, ani defnicji dodatkowych parametrów.

Kryterium GCV jest wskaźnikiem określającym przybliżenie minimum predykcyjnego błędu średniokwadratowego (ang. *predictive mean square error*) zdefiniowanego jako

$$T(\lambda) = \frac{1}{N}\sum_{i=1}^{N}(s_\lambda(\mathbf{x}_i) - g(\mathbf{x}_i))^2,$$

który zależy od estymowanej funkcji g.

Niech $A(\lambda)^\# = (A^TA+\lambda I)^{-1}A^T$ oznacza zregularyzowaną macierz odwrotną, gdzie A jest macierzą interpolacyjną. Wartość oczekiwaną błędu predykcyjnego $T(\lambda)$ można zapisać jako sumę obciążenia i wariancji:

$$\begin{aligned} ET(\lambda) &= \frac{1}{N}E\left(||A(\lambda)^\#(g+\epsilon) - g||^2\right) \\ &= \frac{1}{N}||(I - A(\lambda)^\#)g||^2 + \frac{\sigma^2}{N}\mathrm{Tr}\left[A(\lambda)^\#\right]^2 \\ &= b^2(\lambda) + \sigma^2\mu_2(\lambda), \end{aligned}$$

gdzie $b(\lambda)$ jest obciążeniem modelu, a σ^2 wariancją, natomiast $\mu_2(\lambda)$ jest pewną funkcją wartości szczególnych macierzy A (zob. [14], [29]).

Dla zagadnienia (2.5) kryterium GCV zdefiniowane jest jako

$$V(\lambda) = \frac{1}{N}\sum_{k=1}^{N}\frac{(y_k - s_\lambda(\mathbf{x}_i))^2}{1-\mu_1(\lambda)},$$

gdzie

$$\mu_1(\lambda) = \frac{1}{N}\sum_{i=1}^{N}a_{ii}(\lambda) = \frac{1}{N}\,\mathrm{Tr}\left[A(\lambda)^\#\right],$$

a a_{ii} jest elementem diagonalnym macierzy $A(\lambda)^\#$.

W postaci macierzowej $V(\lambda)$ możemy zapisać

$$V(\lambda) = \frac{||(I - A(\lambda)^\#)y||^2}{[\mathrm{Tr}(I - A(\lambda)^\#)]^2}. \tag{2.44}$$

Słabe twierdzenie o zbieżności kryterium GCV (zob. [29]) mówi, że dla $N \to \infty$ istnieje ciąg $\tilde{\lambda}_N$ wartości λ minimalizujący wartość oczekiwaną kryterium GCV

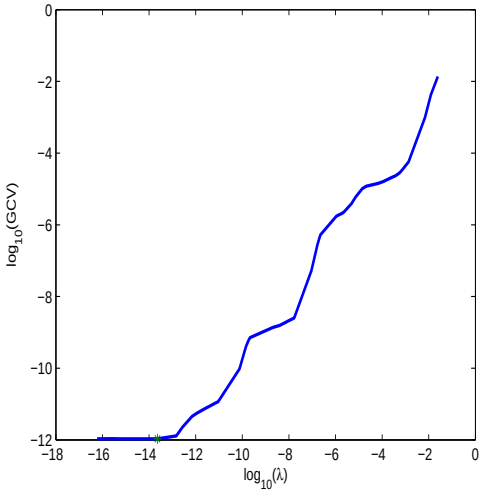
$$EV(\lambda) = \frac{b^2(\lambda) + \sigma^2(1 - 2\mu_1(\lambda) + \mu_2(\lambda))}{(1+\mu_1(\lambda))^2},$$

które przybliżają wartość λ dającą minimum predykcyjnego błędu średniokwadratowego

$$\lambda = \arg\min_\lambda ET(\lambda).$$

Dla danych nieregularnych i wolnych od błędów pomiaru metoda ta zawsze daje małe λ, które stabilizuje obliczanie odwrotności macierzy interpolacyjnej. Mniejsze wartości λ powodują powstanie błędu wynikającego z błędów zaokrągleń.

Na rysunku 2.6 pokazano przykładowy wykres wartości wskaźnika GCV względem λ.



Rysunek 2.6. Wykres wskaźnika $GCV(\lambda)$ dla przykładowego zbioru 30 punktów danych z dwuwymiarowego procesu optymalizacyjnego

Metoda L-krzywej

Drugą metodą wyboru parametru regularyzacji jest metoda L–krzywej zaproponowana przez Hansena [14], jako narzędzie w rozwiązaniu zagadnień źle postawionych metodą regularyzacji Tichonowa.

Metoda ta polega na lokalizacji optymalnego parametru regularyzacji λ na wykresie normy $||L\mathbf{w}_\lambda||_2$ względem normy residuum $||A\mathbf{w}_\lambda - y||_2$. Krzywa ta zdefiniowana jest na skali logarytmicznej jako

$$(\zeta(\lambda), \eta(\lambda)) = (\log ||A\mathbf{w}_\lambda - y||_2, \log ||L\mathbf{w}_\lambda||_2)$$

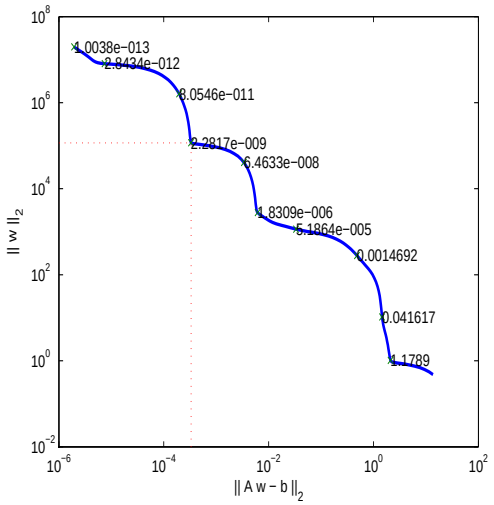
i określa wkład tych wielkości w rozwiązaniu $\mathbf{w}_\lambda$ w zależności od λ.

Rozwiązanie zagadnienia (2.42) dla dowolnego λ leży na tej krzywej. Rozwiązania dla małego λ znajdują się w górnej części wykresu ponad „narożnikiem" wykresu. Jeśli λ jest duże, to rozwiązanie znajduje się w prawej dolnej części wykresu. Optymalny wybór parametru λ określa rozwiązanie znajdujące się

w narożniku wykresu. Narożnik wykresu zdefiniowany jest jako punkt o maksymalnej krzywiźnie

$$\kappa(\lambda) = \frac{\zeta''(\lambda)\eta''(\lambda) - \zeta''(\lambda)\eta'(\lambda)}{((\zeta'(\lambda))^2 + (\eta'(\lambda))^2)^{3/2}}.$$

Maksymalizacja funkcji $\kappa(\lambda)$ prowadzi do algorytmu wyboru wartości parametru regularyzacji λ, dla którego zachowany jest balans pomiędzy $||A\mathbf{w}_\lambda - y||_2$ i $||L\mathbf{w}_\lambda||_2$.



Rysunek 2.7. L–krzywa dla macierzy interpolacyjnej dla funkcji Gaussa i dla zbioru 30 punktów danych z dwuwymiarowego procesu optymalizacyjnego

Obliczenie L–krzywej wymaga przeskanowania spektrum wartości szczególnych macierzy A, co jest operacja czasochłonną. Algorytmy lokalizacji narożnika na wykresie opierają się na następującym schemacie:

1. Rozpocząć z kilkoma punktami (ζ_i, η_i) po obu stronach narożnika, tzn. dla dużego i dla małego λ.
2. Obliczyć trójparametryczną krzywą sklejaną trzeciego stopnia $\mathcal{S}$ dla punktów $(\zeta_i, \eta_i, \lambda_i)$.
3. Niech $\mathcal{S}_2$ oznacza pierwsze dwie współrzędne na krzywej $\mathcal{S}$, takie że $\mathcal{S}_2$ przybliża L–krzywą. Obliczyć punkt na $\mathcal{S}_2$ o maksymalnej krzywiźnie i odczytać odpowiadającą wartość λ_0.
4. Rozwiązać problem regularyzacyjny dla $\lambda = \lambda_0$ oraz dodać nowy punkt $(\zeta(\lambda_0), \eta(\lambda_0), \lambda_0)$ do zbioru węzłów krzywej $\mathcal{S}$.
5. Przejść od punktu 2, jeśli nie uzyskaliśmy zbieżności.

Dla zagadnień źle postawionych w postaci całki Fredholma, dla których spełniony jest warunek Picarda, metoda L–krzywej daje wyniki zbliżone do metody GCV. Dla zagadnień interpolacyjnych oraz aproksymacyjnych dla małych zbiorów danych, które nie są zaburzone, takich jakie otrzymujemy z algorytmu optymalizacyjnego, metoda L–krzywej daje wyniki gorsze niż metoda GCV. Wynika to z tego, że L–krzywa może zawierać co najmniej kilka punktów o dużej krzywiźnie. Pokazano to na rysunku 2.7.

Metoda ważonej wariancji gradientu

Wiadomo, że w wyrażeniu $||A\mathbf{w} - y||^2 + \lambda^2||L\mathbf{w}||_2^2$ dla $\mathbf{w} = \mathbf{w}_\lambda$ w zależności od wartości parametru regularyzacji λ w różny sposób zachowują się składniki $||A\mathbf{w}_\lambda - y||^2$ (składowa rezydualna) i $||L\mathbf{w}_\lambda||_2^2$ (składowa regularyzacyjna). Gdy $\lambda \to 0$, maleje część rezydualna, natomiast część regularyzacyjna rośnie i odwrotnie, wraz ze wzrostem wartości parametru λ rośnie część rezydualna i maleje część regularyzacyjna.

Metoda dyskrepancyjna wyboru parametru λ polega na wybraniu największego λ, dla którego

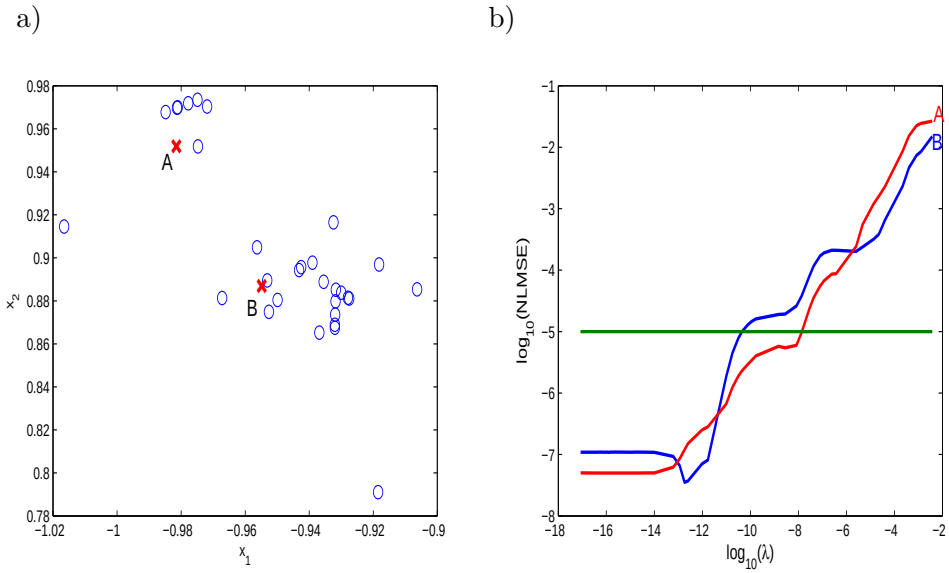
$$||A\mathbf{w}_\lambda - y||^2 \leqslant \delta_e,$$

gdzie δ_e jest ustalonym przez użytkownika poziomem odtwarzania całego zbioru danych. Oczywiście, gdy dane w zbiorze zawierają błąd, wartość δ_e powinna odzwierciedlać wiedzę o poziomie błędu danych. W metodzie tej pod uwagę brane są jednakowo wszystkie punkty danych zawarte w zbiorze $\mathbf{X}$.

Rozpatrzmy proces treningu sieci neuronowej $s_\lambda(\mathbf{x})$ dla kolejnych malejących wartości parametru λ za pomocą rozwiązania zagadnienia regularyzacyjnego dla nieregularnego zbioru danych. Okazuje się, że w pewnych regionach otoczki wypukłej zbioru danych, gdy λ maleje, odtwarzanie zbioru danych w otoczeniu punktu wyznaczania wartości maleje szybciej niż w innych regionach otoczki wypukłej zbioru danych.

W zastosowaniu sieci neuronowej do aproksymacji funkcji celu w procesach optymalizacyjnych praktycznie interesuje nas aproksymacja tylko w bliskim otoczeniu punktu wartościowania. Tak więc parametr λ możemy wybrać w zależności od lokalnego odtwarzania wartości funkcji na zbiorze danych $\mathbf{X}$.

W takim duchu zaproponowana została metoda WGV (zob. [4]), która jest lokalną odmianą metody dyskrapancyjnej. W metodzie tej rozwiązuje się ciąg zagadnień regularyzacyjnych dla malejącej wartości parametru λ. Wyboru parametru λ dokonujemy spośród tych wartości, dla których lokalnie jest uzyskane odtwarzanie zbioru danych na określonym przez użytkownika poziomie. Lokalne



Rysunek 2.8. a) zbiór 30 punktów otrzymanych ze ścieżki optymalizacyjnej dla funkcji Rosenbrocka dwóch zmiennych, b) lokalne odtwarzanie zbioru treningowego w otoczeniu punktu A oraz punktu B

odtwarzanie zbioru danych $\mathbf{X}$ (*Normalized Local Mean Square Error*) definiujemy przez wskaźnik

$$NLMSE_{\lambda,\mathbf{Z}}(\mathbf{x}) = \Big(\sum_{k=1}^{N} \frac{[s_\lambda(\mathbf{x}_k) - y_k]^2}{y_k^2 r_k^2} \Big/ \sum_{k=1}^{N} \frac{1}{r_k^2} \Big)^{\frac{1}{2}}, \tag{2.45}$$

gdzie $r_k = ||\mathbf{x}_k - \mathbf{x}||$. Wybór λ ograniczamy do takiego zbioru $\Lambda(\mathbf{x})$ wartości parametru λ, dla których $NLMSE_{\lambda,\mathbf{Z}} \leqslant \delta_e$. Fakt, że zbiór $\Lambda(\mathbf{x})$ rzeczywiście zależy od punktu wartościowania $\mathbf{x}$ pokazujemy na przykładzie zbioru danych otrzymanym podczas konstrukcji sieci aproksymującej w algorytmie optymalizacji bezgradientowej dwuwymiarowej funkcji Rosenbrocka (rys. 2.8). Na rysunku 2.8a przedstawiono zbiór danych, a na rysunku 2.8b wykresy wskaźnika $NLMSE$ dla punktów z otoczenia punktu A oraz dla punktu B w zależności od wartości parametru λ. Jak widać, odtwarzanie zbioru danych na poziomie błędu względnego 10^{-5} dla otoczenia punktu A otrzymujemy na zbiorze $\Lambda(A) \approx (0, 10^{-8})$. Dla punktu B mamy natomiast przedział $\Lambda(B) \approx (0, 10^{-11})$.

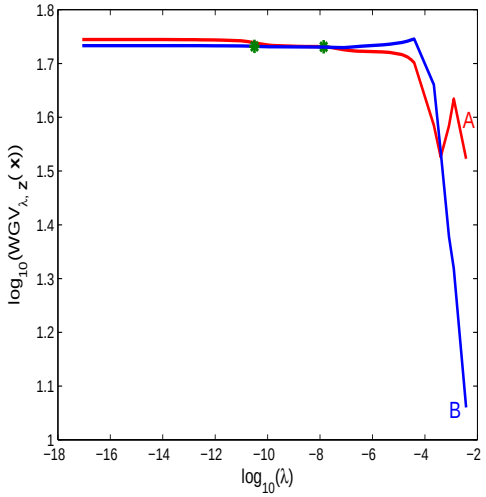
Ze zbioru $\Lambda(\mathbf{x})$ wybieramy takie λ, które minimalizuje funkcjonał ważonej wariancji gradientu zdefiniowanej jako

$$WGV_{\lambda,\mathbf{Z}}(\mathbf{x}) = \sum_{k=1}^{N} \frac{||\nabla s_\lambda(\mathbf{x}_k) - \vec{G}_\lambda(\mathbf{x})||_2^2}{r_k^2} \Big/ \sum_{k=1}^{N} \frac{1}{r_k^2}, \tag{2.46}$$

gdzie $\vec{G}_\lambda(\mathbf{x})$ jest uśrednionym gradientem w punkcie $\mathbf{x}$, określonym jako

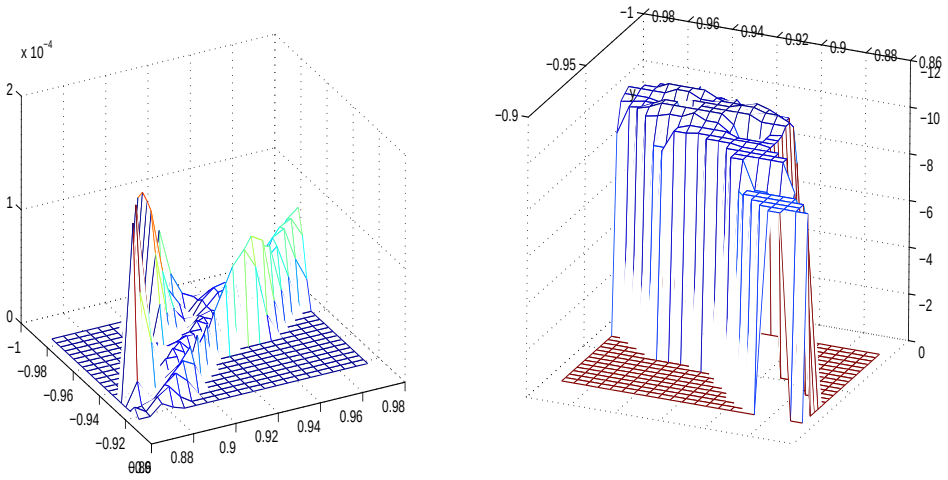
$$\vec{G}_\lambda(\mathbf{x}) = \sum_{k=1}^{N} \frac{\nabla s_\lambda(\mathbf{x}_k)}{r_k^2} \Big/ \sum_{k=1}^{N} \frac{1}{r_k^2} \tag{2.47}$$

oraz $r_k = ||\mathbf{x}_k - \mathbf{x}||_2$. $\vec{G}_\lambda(\mathbf{x})$ oraz $WGV_{\lambda,\mathbf{Z}}(\mathbf{x})$ zależą od $\mathbf{x}$ jedynie przez skalowanie, które wzmacnia wkład punktów $\mathbf{x}_k$ najmniej oddalonych od punktu $\mathbf{x}$. Minimum funkcjonału $WGV_{\lambda,\mathbf{Z}}(\mathbf{x})$ względem parametru $\lambda \in \Lambda(\mathbf{x})$ określa ten model, dla którego $s_\lambda(\mathbf{x})$ ma najmniejsze oscylacje w otoczeniu punktu $\mathbf{x}$, gdyż odchylenie gradientu od uśrednionego gradientu $\vec{G}_\lambda(\mathbf{x})$ dla punktów $\mathbf{x}_k$ najbliższych punktowi $\mathbf{x}$ jest najmniejsze. Na rysunku 2.9 pokazane są wykresy funkcjonału $WGV_{\lambda,\mathbf{Z}}(\mathbf{x})$, odpowiadając wykresom $NLMSE_{\lambda,\mathbf{Z}}(\mathbf{x})$ na rysunku 2.9b dla punktów położonych w otoczeniu dwóch różnych punktów ze zbioru $\mathbf{Z}$.



Rysunek 2.9. Wykres wskaźnika $WGV_{\lambda,\mathbf{Z}}(\mathbf{x})$ dla punktu A oraz punktu B z rysunku 2.8a. Symbolem $*$ na wykresie oznaczona jest wartość λ wskazana przez metodę, dla $NLMSE_{\lambda,\mathbf{Z}}(\mathbf{x}) \leqslant 10^{-4}$

a) b)



Rysunek 2.10. a) Błąd aproksymacji dla λ wybranego przez wskaźnik WGV dla wartości progowej $NLMSE_{\text{prog}} = 5.0 \cdot 10^{-6}$. b) Wybrana wartość λ – jak widać, w regionie uboższym w dane wskaźnik daje wiekszą wartość parametru λ

Jak widać, lokalność tych wskaźników jest uzyskiwana dzięki skalowaniu składnika sumy dla i-tego punktu przez kwadrat jego odległości od punktu wartościowania.

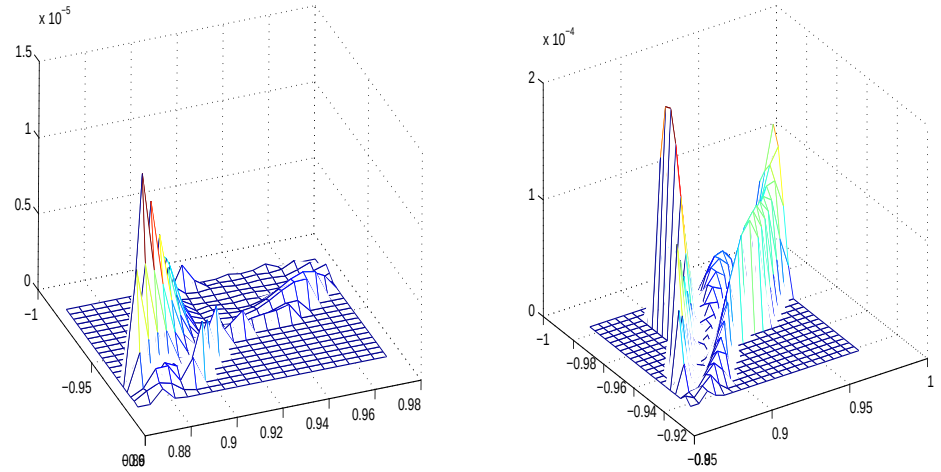
2.3. Optymalizacja – aproksymacja funkcji celu

Rozważmy zadanie minimalizacji nieliniowej funkcji wypukłej $g : \mathbb{R}^n \to \mathbb{R}$

$$\min_{\mathbf{x} \in \mathbb{R}^n} g(x).$$

Załóżmy, że $g \in C^2$ oraz że wyznaczanie wartości funkcji g jest bardzo czasochłonne, np. wymaga zasymulowania pewnego procesu fizycznego. Zasadne jest w takiej sytuacji użycie w procesie optymalizacyjnym aproksymacji funkcji g w celu skrócenia czasu wykonania procesu optymalizacji. W tym podrozdziale omówimy dwa podejścia do przyspieszania optymalizacji funkcji przez zastąpienie jej bezpośredniego czasochłonnego wyznaczania wartości funkcji celu jej aproksymacją. Pierwszą klasą metod są metody regionu wiarygodności (ang. *trust region methods*). Szeroki przegląd tych metod wraz z rysem historycznym znaleźć można w [8].

a) b)



Rysunek 2.11. Błąd aproksymacji dla $\lambda \approx 10^{-13}$ wybranego przez wskaźnik GCV (a). Błąd aproksymacji dla $\lambda \approx 10^{-8.5}$ wybranego za pomocą L–krzywej (b).

W metodach regionu wiarygodności buduje się sekwencję modeli, aproksymujących funkcję celu, skonstruowanych na danych uzyskanych z algorytmu próbkującego. Model w takiej sekwencji jest optymalizowany algorytmem gradientowym w regionie wiarygodności będącym otoczeniem ostatniego punktu, w którym wartość funkcji celu była wyznaczona w sposób bezpośredni. W znalezionym minimum modelu jest wykonywane kolejne wartościowanie bezpośrednie funkcji celu oraz modyfikowany jest promień regionu wiarygodności.

Drugim omawianym w tym paragrafie podejściem jest metoda zaprezentowana w [4]. W metodzie tej optymalizuje się funkcję g algorytmem bezgradientowym, a model aproksymujący konstruowany jest dla kolejnego punktu, w którym wartość funkcji celu ma być wyznaczona w sposób bezpośredni, jeśli punkt znajduje się w regionie, w którym możliwe jest skonstruowanie wiarygodnego modelu aproksymującego. W obydwu omawianych podejściach jako model aproksymujący użyte mogą być sieci z radialnymi funkcjami bazowymi ([4], [21]).

2.3.1. Metoda regionu wiarygodności

Niech $g : \mathbb{R}^n \to \mathbb{R}$ będzie funkcją celu procesu optymalizacyjnego. Niech ponadto $g \in C^2$. Schematycznie metody regionu wiarygodności można przedstawić następująco:

1. **Inicjalizacja**
   Dane są: punkt $\mathbf{x}_0 \in \mathbb{R}^n$ oraz początkowy region wiarygodności o promieniu $\Delta_0 \in \mathbb{R}$, a także stałe sterujące $\eta_1, \eta_2, \gamma_1, \gamma_2 \in \mathbb{R}$ takie, że
   $$0 < \eta_1 \leqslant \eta_2 < 1, \qquad 0 < \gamma_1 \leqslant \gamma_2 < 1. \tag{2.48}$$
   Obliczyć wartość $g(\mathbf{x}_0)$ i ustawić $k = 0$.
2. **Konstrukcja modelu**
   Skonstruować model s_k aproksymujący funkcję celu g w regionie wiarygodności $B(\mathbf{x}_k, \Delta_k) = \{\mathbf{x} : ||\mathbf{x} - \mathbf{x}_k|| \leqslant \Delta_k\}$ dla zbioru $\mathbf{Z^{(k)}}$ danych $n+1$ punktów wygenerowanych przez algorytm bezpośredni.
   a) Przed konstrukcją modelu aproksymującego dokonać oceny czy model dla zbioru $\mathbf{Z}$ będzie modelem poprawnym.
   b) Jeśli zbiór $\mathbf{Z}$ nie zapewnia poprawności modelu, to wymienić pewną liczbę punktów w zbiorze $\mathbf{Z}$ na punkty z poprzednich iteracji algorytmu bezpośredniego lub wykonać nową iterację algorytmu bezpośredniego i dołączyć punkt powstały w jej wyniku do zbioru $\mathbf{Z}$.
3. **Obliczanie kroku**
   Obliczyć krok $\mathbf{d}_k$, który „zapewni odpowiednią redukcję modelu" s_k taki, że $\mathbf{x}_k + \mathbf{d}_k \in B(\mathbf{x}_k, \Delta_k)$.
4. **Akceptacja nowego punktu**
   Obliczyć $g(\mathbf{x}_k + \mathbf{d}_k)$, a następnie wskaźnik redukcji modelu
   $$\rho_k = \frac{g(\mathbf{x}_k) - g(\mathbf{x}_k + \mathbf{d}_k)}{s_k(\mathbf{x}_k) - s_k(\mathbf{x}_k + \mathbf{d}_k)} \tag{2.49}$$
   Jeśli $\rho_k \geqslant \eta_1$, to $\mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{d}_k$, w przeciwnym wypadku $\mathbf{x}_{k+1} := \mathbf{x}_k$.
5. **Modyfikacja promienia wiarygodności**
   Ustawić
   $$\Delta_{k+1} \in \begin{cases} [\Delta_k, +\infty), & \text{dla } \rho_k > \eta_2, \\ (\gamma_2 \Delta_k, \Delta_k) & \text{dla } \rho_k \in [\eta_1, \eta_2), \\ (\gamma_1 \Delta_k, \gamma_2 \Delta_k], & \text{dla } \rho_k < \eta_1. \end{cases} \tag{2.50}$$
6. Inkrementuj k i przejdź do kroku 2.

Minimalizacja modelu aproksymującego

Kluczowymi krokami w tym schemacie są punkty 2 i 3. W punkcie 2 budujemy model aproksymujący funkcję g w kuli $B(\mathbf{x}_k, \Delta_k)$. Klasycznie jako metoda aproksymacji użyty może być model kwadratowy

$$s_k(\mathbf{x}) = g(\mathbf{x}_k) + \nabla g(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T H_k(\mathbf{x} - \mathbf{x}_k),$$

gdzie H_k jest aproksymacją Hesjanu funkcji g. W pracy [21] wykorzystany jest natomiast model sieci z radialnymi funkcjami bazowymi

$$s_k(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(||\mathbf{x} - \mathbf{x}_i||) + \sum_{i=1}^{Q} \beta_i p_i(x),$$

gdzie $p_i \in \pi_m(\mathbb{R}^n)$ są wielomianami bazy przestrzeni wielomianów n zmiennych stopnia nie większego niż m, gdzie m jest stopniem warunkowej dodatnio określoności funkcji radialnej ϕ a $Q = \binom{m+n-1}{n}$.

Poprawność modelu aproksymującego

Pojęcie poprawności modelu (ang. *validity of the model*) definiujemy następująco:

Definicja 13
Mówimy, że dla danej metody regionu wiarygodności zbiór $\mathbf{Z}^{(k)}$ zbudowany dla k-tej iteracji zapewni poprawność modelu aproksymującego s_k, jeśli dla każdego
k prawdziwe będzie ograniczenie gradientu

$$||\nabla s_k(\mathbf{x}) - \nabla g(\mathbf{x})|| \leqslant \frac{1}{(n+1)!} G \Lambda_{\mathbf{X}^{(k)}} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{x}||^{q+1}$$

dla dowolnego $\mathbf{x} \in B(\mathbf{x}_k, \Delta_k)$, gdzie q jest najwyższym stopniem wielomianu interpolacyjnego n zmiennych, jaki można zbudować dla danych ze zbiorze $\mathbf{Z}^{(k)}$ i ciąg stałych $\Lambda_{\mathbf{X}^{(k)}}$ jest ograniczony z góry.

Istnienie górnego ograniczenia ciągu $\{\Lambda_{\mathbf{X}^{(k)}}\}_{k=1,\ldots}$ zapewnia utrzymanie takiej geometrii zbioru $\mathbf{X}^{(k)}$ dla każdego kroku k, że zbiór $\mathbf{X}^{(k)}$ będzie zbiorem dobrze wyważonym (ang. *well poised*).

Definicja 14
Mówimy, że zbiór $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_Q\}$ jest wyważony względem przestrzeni wielomianów $\pi_q(\mathbb{R}^n)$, jeśli macierz interpolacyjna, tzn.

$$P = \begin{pmatrix} p_0(\mathbf{x}_1) & \ldots & p_Q(\mathbf{x}_1) \\ \vdots & & \vdots \\ p_0(\mathbf{x}_Q) & \ldots & p_Q(\mathbf{x}_Q) \end{pmatrix}, \tag{2.51}$$

gdzie $Q = \binom{q+n}{n}$ jest nieosobliwa.

Jeśli zbiór $\mathbf{X}$ jest przeskalowany tak, że zawiera się w kuli $B(0,1)$, a jako bazę wybierzemy przeskalowane wielomiany bazy naturalnej, tzn.

$$\{1, x_1, x_2, \ldots, x_d, x_1^2/2, x_1x_2/2, \ldots, x_{n-1}^{r-1}x_n/(q-1)!, x_n^r/q!\}, \tag{2.52}$$

to im większy wyznacznik macierzy (2.51), tym zbiór jest lepiej wyważony. Sprawdzenie wyważenia zbioru $\mathbf{X}$ z powyższej definicji nie daje nam informacji, które z punktów zbioru $\mathbf{X}$ degradują dobre wyważenie. Nie mamy więc podpowiedzi, który z punktów należy wymienić, aby poprawić wyważenie. Sprawdzenie wyważenia oparte na analizie zachowania fundamentalnych wielomianów Lagrange'a [9] lub fundamentalnych wielomianów Newtona [21], daje nam taką podpowiedź.

Dla dowolnego zbioru $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_Q\}$, który jest $\pi_q(\mathbb{R}^n)$-unisolwentny oraz dla wektora $\mathbf{p}(\mathbf{x}) = [p_0(\mathbf{x}), \ldots, p_Q(\mathbf{x})]^T$, gdzie $\{p_i\}_{i=0}^Q$ jest bazą przestrzeni $\pi_q(\mathbb{R}^n)$, istnieje wektor $\mathbf{u} = [u_0(\mathbf{x}), \ldots, u_Q(\mathbf{x})]$ taki, że

$$\sum_{i=1}^{Q} u_i(\mathbf{x})\mathbf{p}(\mathbf{x}_i) = \mathbf{p}(\mathbf{x}).$$

Funkcje $u_i,\ i = 0, \ldots, Q$ są funkcjami Lagrange'a zmiennej $\mathbf{x}$, tzn. $u_i(\mathbf{x}_j) = \delta_{ij}$. Są to wielomiany, gdyż liniowo zależą od wektora $\mathbf{p}(\mathbf{x})$.

Definicja 15
Mówimy, że zbiór $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_Q\}$ jest Λ-wyważony, jeśli dla dowolnego punktu $\mathbf{x}$ z otoczki wypukłej zbioru $\mathbf{X}$ zachodzi

$$||\mathbf{u}(x)||_1 \leqslant \Lambda.$$

Twierdzenie 16
Jeśli macierz P jest nieosobliwa oraz $||P^{-1}|| \leqslant \Lambda$, to zbiór $\mathbf{X}$ jest $(\sqrt{Q-1}\cdot\Lambda)$-wyważony w kuli $B(0,1)$. Odwrotnie, jeśli zbiór $\mathbf{X}$ jest Λ-wyważony w kuli $B(0,1)$, to macierz P jest nieosobliwa i

$$||P^{-1}|| \leqslant \theta\Lambda,$$

gdzie $\theta > 0$ zależy tylko od n i r, nie zależy natomiast od $\mathbf{X}$ i Λ.

W pracy [9] rozważane są ograniczenia bazujące na rozkładzie LU macierzy P oraz rozkładzie QR macierzy P^T, za pomocą których można zapewnić ograniczenie

$$||P^{-1}|| \leqslant \frac{c(Q)\varepsilon_{\text{growth}}}{\zeta},$$

gdzie $\varepsilon_{\text{growth}}$ jest estymatą czynnika wzrostu związanego z algorytmem faktoryzacji, ζ jest ograniczeniem dolnym modułu obrotów (ang. *pivots*) w faktoryzacji oraz $c(Q)$ jest pewną potęgą liczby $Q-1$. Wówczas, korzystając z twierdzenia 16, otrzymujemy ograniczenie

$$\sum_{i=1}^{Q} |u_i(\mathbf{x})| \leqslant \frac{(Q-1)c(Q)\varepsilon_{\text{growth}}}{\zeta}. \tag{2.53}$$

Twierdzenie 17 (moduły obrotów w rozkładzie LU)
Niech $\zeta \in (0, 1/4)$ będzie ustalone. Za pomocą rozkładu LU można wyznaczyć $Q = (n+1)(n+2)/2$ punktów w kuli $B(0,1)$, zakładając, że $\mathbf{x}_1 = 0$, dla których moduły obrotów w procedurze eliminacji Gaussa wyznaczającej rozkład $P = LDU$ spełniają

$$|D_{ii}| \geqslant \zeta, \quad i = 1, \ldots, Q-1. \tag{2.54}$$

Mamy wówczas

$$||P^{-1}|| \leqslant \frac{\sqrt{Q-1}||L^{-1}||||U^{-1}||}{\zeta}.$$

W nierówności (2.53) mamy więc $\varepsilon_{\text{growth}} = ||L^{-1}||||U^{-1}||$ oraz $c(Q) = \sqrt{Q-1}$.

Twierdzenie 18
Niech $\zeta \in (0, 1/4)$ będzie ustalone. Za pomocą rozkładu LU można wyznaczyć $Q = (n+1)(n+2)/2$ punktów w kuli $B(0,1)$ zakładając, że $\mathbf{x}_1 = 0$, dla których elementy diagonalne macierzy R w rozkładzie QR macierzy P^T spełniają

$$|R_{ii}| \geqslant \frac{\zeta}{\sqrt{Q-1}}, \quad i = 1, \ldots, Q-1. \tag{2.55}$$

Mamy wówczas

$$||P^{-1}|| \leqslant \frac{(Q-1)||\bar{R}^{-1}||}{\zeta},$$

gdzie $P^T = QR = QD\bar{R}$ i $\bar{R} \in \mathbb{R}^{(Q-1)\times(Q-1)}$ jest macierzą trójkątną górną z jedynkami na przekątnej. W nierówności (2.53) mamy więc $\varepsilon_{\text{growth}} = ||\bar{R}^{-1}||$ oraz $c(Q) = Q-1$.

Drugą metodą sprawdzenia wyważenia zbioru $\mathbf{X}$ oraz wykrycia punktów, które degradują wyważenie, jest analiza modułów obrotów w algorytmie konstrukcji fundamentalnych wielomianów Newtona [8]. W metodzie tej w kroku startowym kolejnym punktom zbioru $\mathbf{X}$ przyporządkowane są kolejne wielomiany bazy naturalnej (2.52). Wynikiem natomiast jest podział zbioru $\mathbf{X}$ na $q+1$ bloków

$$\mathbf{X}^{[l]} = \{\mathbf{x}_1^{[l]}, \ldots, \mathbf{x}_{|\mathbf{Y}^{[l]}|}^{[l]}\} \quad (l = 0, \ldots, r),$$

gdzie l-ty blok zawiera

$$|\mathbf{X}^{[l]}| = \binom{l+r-1}{l}$$

punktów. Każdemu punktowi $\mathbf{x}_i^{[l]} \in \mathbf{X}^{[l]}$ odpowiada fundamentalny wielomian Newtona stopnia l, spełniający warunki:

$$N_i^{[l]}(\mathbf{x}_j^{[m]}) = \delta_{ij}\delta_{lm} \quad \text{dla każdego } \mathbf{x}_j^{[m]} \in \mathbf{X}^{[m]} \quad \text{dla } m \leqslant l.$$

Algorytm 1 (wyznaczania bazy fundamentalnych wielomianów Newtona)

1. **Inicjalizacja**
   Ustawić $N_i^{[l]}(i = 1, \ldots, |\mathbf{X}_{[l]}|, l = 0, \ldots, q)$ *na bazę naturalną (2.52). Ustawić* $\mathbf{X}_{\text{temp}} = \emptyset$.
2. **Pętla główna**
   Dla $l = 0, \ldots, n$, *i* $i = 1, \ldots, |\mathbf{X}^{[l]}|$,
   a) wybrać punkt $\mathbf{x}_i^{[m]} \in \mathbf{X} \backslash \mathbf{X}_{\text{temp}}$ *taki, że* $|N_i^{[l]}(\mathbf{x}_i^{[l]})| \neq 0$,
   b) jeśli nie istnieje taki punkt $\mathbf{x}_i^{[l]}$ *w zbiorze* $\mathbf{X} \backslash \mathbf{X}_{\text{temp}}$, *to ustawić* $\mathbf{X} = \mathbf{X}_{\text{temp}}$ *i zakończyć przedwcześnie (baza wielomianów Newtona jest niepełna).*
   c) dodać punkt $\mathbf{x}_i^{[l]}$

$$\mathbf{X}_{\text{temp}} \leftarrow \mathbf{X}_{\text{temp}} \cup \{\mathbf{x}_i^{[l]}\},$$

   d) znormalizować odpowiadający wielomian fundamentalny

$$N_i^{[l]}(\mathbf{x}) = N_i^{[l]}(\mathbf{x}) / |N_i^{[l]}(\mathbf{x}_i^{[l]})|, \tag{2.56}$$

   e) uaktualnić odpowiadające wielomiany fundamentalne w bloku l

$$N_j^{[l]}(\mathbf{x}) = N_j^{[l]}(\mathbf{x}) - N_j^{[l]}(\mathbf{x}_i^{[l]}) N_i^{[l]}(\mathbf{x}),$$

   dla $j \neq i, j = 1, \ldots, |\mathbf{X}^{[l]}|$, i

$$N_j^{[k]}(\mathbf{x}) = N_j^{[k]}(\mathbf{x}) - N_j^{[k]}(\mathbf{x}_i^{[l]}) N_i^{[l]}(\mathbf{x}),$$

   dla $j = 1, \ldots, |\mathbf{X}^{[l]}|, k = l+1, \ldots, q$.

Algorytm podany jest algorytmem ortogonalizacji Gramma–Schmidta wyjściowej bazy naturalnej względem iloczynu skalarnego

$$\langle P, Q \rangle = \sum_{i=1}^{Q} P(\mathbf{x}_i) Q(\mathbf{x}_i).$$

Modułami obrotów w tej procedurze nazywamy wielkości $N_i^{[l]}(\mathbf{x}_i^{[l]})$ w podstawieniu (2.56).

Twierdzenie 19
Algorytm 1 generuje zbiór $\mathbf{X}_{\text{temp}}$*, który jest wyważony, jeśli moduły obrotów* $N_i^{[i]}(\mathbf{x}_i^{[l]})$ *są różne od zera.*

Dodanie do podanej procedury warunku

$$N_i^{[l]}(\mathbf{x}_i^{[l]}) > \theta, \tag{2.57}$$

dla pewnego ustalonego θ, umożliwia kontrolę jakości wyważenia.

Poprawa zbioru danych

Oceny jakości modelu aproksymującego s_k w k-tej iteracji metody regionu wiarygodności dokonujemy więc za pomocą jednej z podanych metod kontroli wyważenia zbioru $\mathbf{X}^{(k)}$. Dla wielomianów Lagrange'a jest to skorzystanie z twierdzenia 17 lub 18 dla ustalonych wartości progowych modułów obrotów odpowiednio
w nierównościach (2.54) i (2.55). W pracy [9] rekomendowane jest użycie metody QR, czyli twierdzenia 18 i ograniczeń modułów obrotów (2.55) jako metody bardziej stabilnej numerycznie. W pracy [8] rekomendowany jest natomiast algorytm 1 konstrukcji wielomianów Newtona oraz oceny modelu za pomocą nierówności (2.57). W pracy [21] do oceny wyważenia zbioru $\mathbf{X}^{(k)}$ użyty jest algorytm 1 dla $q = 1$.

Korzystając z nierówności (2.54), (2.55) lub (2.57), można więc ocenić, które z punktów zbioru $\mathbf{X}$ degradują dobre wyważenie. Takie punkty mogą być wymienione na inne spoza zbioru $\mathbf{X}$, dla których obliczona została już wartość funkcji celu, a które poprawią wyważenie zbioru danych.

Jeśli skonstruowany już został wiarygodny model w kuli $B(\mathbf{x}_k, \Delta_k)$, to przechodzimy do kroku 3, czyli do minimalizacji modelu w celu wyznaczenia wektora $\mathbf{d}_k$. Ponieważ gradient modelu możemy obliczyć szybko, do znajdowania minimum stosujemy więc dowolny algorytm drugiego rzędu lub jakąś odmianę algorytmu quasi-newtonowskiego.

Zbieżność metod regionu wiarygodności

Założeniami dotyczącymi sekwencji $\{s_0, s_1, \ldots, s_k, \ldots\}$ konstruowanych modeli aproksymujących, dzięki którym można dowieść zbieżności metody regionu wiarygodności, są :

1. Model aproksymujący s_k spełnia

$$|g(\mathbf{x}) - s_k(\mathbf{x})| \leqslant C_1 \Delta_k^2, \quad \text{dla każdego} \quad \mathbf{x} \in B(\mathbf{x}_k, \Delta_k)$$

gdzie C_1 jest niezależne od $\mathbf{x}$ i k.

2. Gradienty zarówno funkcji g, jak i modelu aproksymującego spełniają

$$||\nabla g(\mathbf{x}_k) - \nabla s_k(\mathbf{x}_k)|| \leqslant C_2\Delta.$$

3. Hesjan modelu aproksymującego jest ograniczony w regionie wiarygodności $B(\mathbf{x}_k, \Delta_k)$ dla każdego k

$$||\nabla_{\mathbf{xx}} s_k(\mathbf{x})|| < C \quad \text{dla każdego } \mathbf{x} \in B(\mathbf{x}_k, \Delta_k).$$

4. Dla każdego k mamy

$$s_k(\mathbf{x}_k) - s_k(\mathbf{x}_k + \mathbf{d}_k) \geqslant \kappa ||\nabla s_k(\mathbf{x}_k)|| \min\left(\frac{||\nabla s_k(\mathbf{x}_k)||}{\beta_k}, \Delta_k\right),$$

gdzie $\kappa \in (0,1)$ a

$$\beta_k = 1 + \max_{\mathbf{x} \in B(\mathbf{x}_k, \Delta_k)} ||\nabla_{\mathbf{xx}} s_k(\mathbf{x})||.$$

5. Zwiększ k – przejdź do punktu 2.

O zbieżności metody regionu wiarygodności mówi następujące twierdzenie:

Twierdzenie 20
Jeśli zachodzą warunki 1.–5. to algorytm regionu wiarygodności jest zbieżny do punktu krytycznego pierwszego rzędu $\mathbf{x}^*$*, tzn.* $\nabla g(\mathbf{x}^*) = 0$.

W pracy [21] wyprowadzono wartości stałych C_1 i C_2 dla modelu aproksymującego w postaci sieci z n+1 radialnymi funkcjami bazowymi (2.3). W przykładach numerycznych prezentowanych w pracy [21] użyta jest funkcja bazowa $\phi(r) = r^3$. W rozwinięciu (2.3) występuje więc wielomian liniowy n zmiennych.

2.3.2. Metoda SPELROA

Drugą metodą przyspieszenia procesu optymalizacji przez wykorzystanie aproksymacji funkcji celu jest metoda SPELROA (ang. *Search Procedure Exploiting a Local Regularized Objective Approximation*) zaproponowana przez autora w [3] i rozwinięta w [4]. Polega ona na konstrukcji modelu aproksymującego przed kolejnym wyznaczeniem wartości funkcji celu w punkcie wygenerowanym przez algorytm podstawowy dokonujący optymalizacji. Najczęściej dla funkcji, dla których wyznaczanie wartości jest czasochłonne, jest to algorytm bezgradientowy [10]. Metoda ta zaproponowana została w połączeniu z bezgradientowym algorytmem EXTREM [15] użytym do optymalizacji magnesów zakrzywiających akceleratora LHC (ang. *Large Hadron Collider*) [23].

Metoda prezentowana dalej (zob. [4]) może zostać połączona z dowolnym algorytmem optymalizacyjnym. Najlepsze wyniki, tzn. najlepsze przyspieszenie

procesu optymalizacji, uzyskiwane są jednak dla algorytmów bezgradientowych. Schemat metody jest następujący:

1. Wykonać I_s początkowych kroków podstawowego algorytmu optymalizacyjnego.
2. Algorytmem podstawowym wygenerować punkt $\mathbf{x}_k$, w którym ma być obliczona funkcja $g(\mathbf{x}_k)$.
3. Stworzyć zbiór treningowy $\mathbf{Z}$ z punktów, w których funkcja była obliczona w sposób bezpośredni.
4. Ocenić, czy można dla punktu $\mathbf{x}_k$ skonstruować model aproksymujący, będący wiarygodnym modelem do aproksymacji funkcji g w punkcie $\mathbf{x}_k$:
   a) Jeśli punkt $\mathbf{x}_k$ znajduje się w wiarygodnym regionie dziedziny, to skonstruować model aproksymujący s_k dla punktu $\mathbf{x}_k$.
   Jeśli model został poprawnie skonstruowany, to obliczyć aproksymację $s_k(\mathbf{x}_k)$ i podstawić
   $$g(\mathbf{x}_k) \leftarrow s_k(\mathbf{x}_k).$$
   b) Jeśli punkt $\mathbf{x}_k$ nie jest w regionie wiarygodnym lub model nie może zostać poprawnie skonstruowany, to obliczyć funkcję $g(\mathbf{x}_k)$ w sposób bezpośredni.
5. Zwiększyć k – przejść do kroku 2.

Zakończyć podany proces, gdy spełnione jest kryterium końca algorytmu podstawowego.

Różnica w porównaniu z metodami regionu wiarygodności omawianej w poprzednim podrozdziale jest taka, że o ile w metodzie regionu wiarygodności obliczamy funkcję w sposób bezpośredni w kolejnym punkcie $\mathbf{x}_k$ wygenerowanym przez algorytm, o tyle w prezentowanej tutaj metodzie chcemy uniknąć bezpośredniego obliczania funkcji. Obliczenie funkcji w sposób bezpośredni w punkcie $\mathbf{x}_k$ jest zastępowane konstrukcją modelu s_k oraz obliczeniem $s_k(\mathbf{x}_k)$, jeśli punkt $\mathbf{x}_k$ znajduje się w regionie wiarygodnym, tzn. dostatecznie bogatym w punkty danych.

Kluczowym krokiem w tym schemacie jest punkt 4. Najpierw dokonuje się sprawdzenia, czy punkt $\mathbf{x}_k$ znajduje się w regionie dostatecznie bogatym w dane, tzn. takim, dla którego można skonstruować wiarygodny model aproksymujący. W zastosowaniu podanej metody w [4] w tym celu użyty został lokalny wskaźnik otoczenia $\gamma(\mathbf{x})$ omówiony w podrozdziale 2.2.6. Jak wynika z własności wskaźnika $\gamma(\mathbf{x})$, mierzy on nie tylko liczbę punktów danych w otoczeniu punktu $\mathbf{x}$, ale również to, jak te dane otaczają punkt $\mathbf{x}$. W punkcie 4a) konstruowany jest model aproksymujący funkcję g w punkcie $\mathbf{x}_k$. Jako model aproksymujący w [4] użyta została sieć (2.3) z radialną funkcją bazową $\phi(r) = e^{-r^2/(\alpha[\mathrm{diam}(\mathbf{X})]^2)}$, gdzie $\mathrm{diam}(\mathbf{X})$ jest średnicą zbioru danych, a $\alpha \in [0.5, 1]$. Jest to funkcja ściśle dodatnio określona. Stąd nie występuje część wielomianowa w (2.3). Ponieważ zbiór danych

X jest mały oraz bardzo nieregularny, problem konstrukcji modelu aproksymującego jest problemem źle postawionym. Do konstrukcji modelu użyta została metoda regularyzacji Tichonowa omówiona w podrozdziale 2.2.7 z operatorem regularyzacji $L = I$. Wektor wag **w** w (2.3) obliczany jest z rozwinięcia (2.43). Jako metody wyboru wartości parametru regularyzacji użyto wskaźnika WGV, dzięki czemu możliwe było uzyskanie jakości aproksymacji funkcji celu z błędem względnym na poziomie 10^{-3}. W punkcie 4b dokonuje się bezpośredniego obliczenia $g(\mathbf{x})$ w wypadku, gdy punkt $\mathbf{x}_k$ leży w obszarze, który nie jest wiarygodny lub gdy jest w obszarze wiarygodnym, lecz dla żadnego λ model nie daje dobrego odtwarzania zbioru **Z** w otoczeniu punktu **x** w sensie wskaźnika $NLMSE_{\lambda,\mathbf{Z}}(\mathbf{x})$.

Jak pokazują testy optymalizacji funkcji Rosenbrocka w następnym podrozdziale, jakość aproksymacji funkcji celu, uzyskana modelem z funkcjami Gaussa, jest lepsza niż w przypadku użycia modelu z $\phi(r) = r^3$ z liniowym składnikiem wielomianowym.

2.4. Testy numeryczne

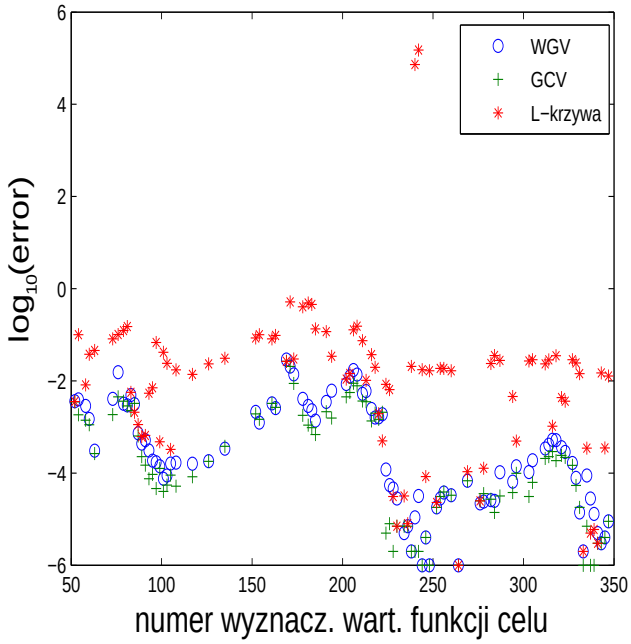
2.4.1. Jakość aproksymacji

Zasadność zastosowania aproksymacji do przybliżania funkcji celu w regionach, w których funkcja celu została wcześniej wyznaczona w sposób bezpośredni w pewnej liczbie punktów, pokazujemy na przykładzie optymalizacji funkcji Rosenbrocka [34] za pomocą algorytmu EXTREM [15]. Na rysunkach 2.12 i 2.13 przedstawiono maksymalny błąd względny aproksymacji funkcji uzyskany na zbiorach testowych 20 punktów wylosowanych z wielowymiarowego rozkładu jednostajnego i spełniających warunek $\gamma(\mathbf{x}) < \gamma_{\text{prog}}$. Z testów dla funkcji Rosenbrocka wynika, że w około 25% punktów wyznaczanie funkcji w sposób bezpośredni może zostać zastąpione przez aproksymację siecią RBF. Wyniki użycia aproksymacji funckji celu zamiast wartościowania bezpośredniego dla funkcji Rosenbrocka przedstawiamy w następnym podrozdziale.

Na rysunku 2.14 przedstawiamy porównanie jakości aproksymacji funkcji celu uzyskanej za pomocą sieci aproksymującej z funkcjami Gaussa z sieciami z funkcjami r^3, takimi jak prezentowane w pracy [21]. Sieć z funkcjami r^3 zachowywała się porównywalnie dobrze tylko na końcu procesu optymalizacji, tzn. blisko punktu minimalnego.

2.4.2. Funkcja testowa

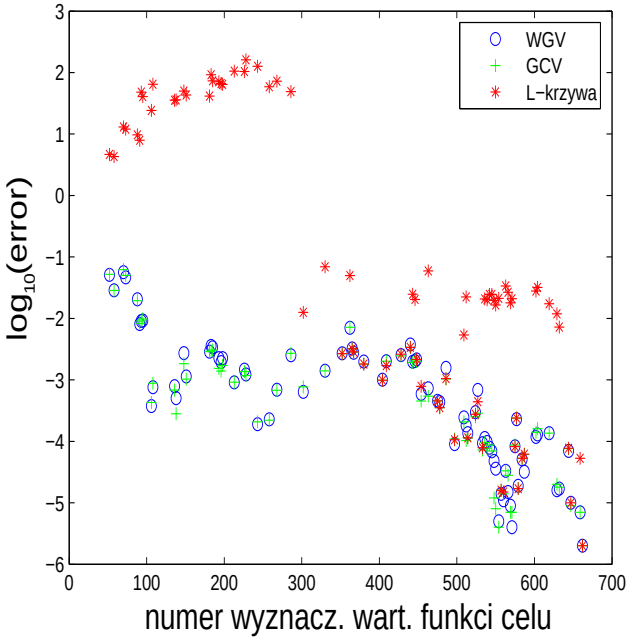
W niniejszym podrozdziale pokazujemy, jakie wyniki daje metoda omówiona w podrozdziale 2.3.2 daje wyniki dla funkcji Rosenbrocka. Metoda w połączeniu



Rysunek 2.12. Maksymalny błąd aproksymacji siecią RBF konstruowaną dla kolejnych punktów, w których wyznaczana jest funkcja celu (dla funkcji Rosenbrocka trzech zmiennych) w kolejnych krokach algorytmu EXTREM [15]. Algorytm wystartowany z punktu $[0, 1, 1]$ i wymagał 347 wyznaczeń wartości funkcji, z których 92 wykonywanych jest w punktach, dla których można zbudować sieć aproksymującą. Wykres pokazuje maksymalny błąd względny dla sieci aproksymującej zbudowanej za pomocą metod GCV, L–krzywej oraz WGV w kolejnych krokach w obszarze, dla którego $\gamma(\mathbf{x}) > 0.55$

z algorytmem bezgradientowym EXTREM [15] użyta została do minimalizacji funkcji trzech i pięciu zmiennych. Tabele 2.1 i 2.2 przedstawiają znalezione minimum oraz liczbę punktów, w których funkcja celu była wyznaczona w sposób bezpośredni oraz liczbę konstrukcji aproksymacji funkcji potrzebnych do znalezienia minimum. Wykresy natomiast przedstawiają wartość funkcji w optymalizacji algorytmem bezpośrednim oraz proponowaną metodą. Wyniki te omówione zostały w prace [6].

Wyniki numeryczne zastosowania metody w procesie optymalizacji rzeczywistych trudnych do wyznaczenia funkcji występujących w optymalizacji magnesów nadprzewodzących przedstawiliśmy w pracy [4].

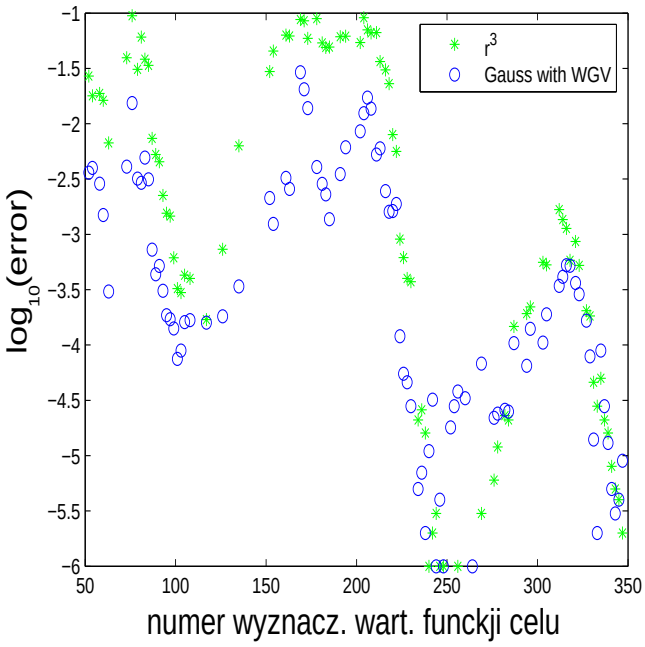


Rysunek 2.13. Ograniczenia górne błędu aproksymacji sieciami skonstruowanymi dla kolejnych punktów, w których funkcja celu jest wyznaczana w sposób bezpośredni (dla funkcji Rosenbrocka pięciu zmiennych) w kolejnych krokach algorytmu EXTREM [15]. Algorytm rozpoczął proces minimalizacji z punktu $[0, 1, 1, 1, 1]$ i wymagał 790 wyznaczeń wartości funkcji, z których 110 wykonywanych jest w punktach, dla których można zbudować sieć aproksymującą. Wykres pokazuje maksymalny błąd względny dla sieci aproksymującej zbudowanej za pomocą metody GCV, L–krzywej oraz WGV w kolejnych krokach w obszarze, dla którego $\gamma(\mathbf{x}) > 0.65$

2.4.3. Optymalizacja magnesów nadprzewodzących

Określenie funkcji celu

Celem optymalizacji magnesów nadprzewodzących w zderzaczu LHC było zapewnienie wymaganych własności generowanego przez nie pola magnetycznego – zakrzywiającego, skupiającego lub korygującego trajektorię wiązki przyspieszanych cząstek elementarnych. Wymagania te są określone przez geometrię tunelu oraz fizykę akceleratora. Ogólnie rzecz biorąc, w procesie konstrukcji magnesu celem jest czyste pole wielobiegunowe o określonym natężeniu. Cel ten formułowany jest w kategoriach współczynników rozwinięcia Fouriera statycznego pola



Rysunek 2.14. Porównanie jakości aproksymacji dla sieci z funkcjami Gaussa konstruowanych metodą WGV oraz sieci z funkcjami r^3 konstruowanych bez regularyzacji (zob. [21]) dla zbiorów danych konstruowanymi dla kolejnych punktów generowanych w optymalizacji funkcji Rosenbrocka trzech zmiennych w kolejnych krokach algorytmu EXTREM [15]. Porównanie dla punktów na ścieżce optymalizacyjnej, dla których $\gamma(\mathbf{x}) > 0.55$

magnetycznego w aperturze magnesu. Statyczne pole magnetyczne w regionach niemagnetycznych i wolnych od prądu jest opisywane przez układ uproszczonych równań Maxwella

$$\begin{aligned} \nabla \times \vec{B} &= 0, \\ \nabla \cdot \vec{B} &= 0. \end{aligned} \tag{2.58}$$

Dla zagadnień dwuwymiarowych układ (2.58) we współrzędnych kartezjańskich redukuje się do układu

$$\frac{\partial B_x}{\partial y} = \frac{\partial B_y}{\partial x},$$

$$\frac{\partial B_x}{\partial z} = -\frac{\partial B_y}{\partial y},$$

Tabela 2.1. Porównanie znalezionego minimum oraz liczby wyznaczeń wartości funkcji celu dla algorytmu EXTREM oraz algorytmu EXTREM połączonego z aproksymacją funkcji celu za pomocą sieci RBF dla funkcji Rosenbrocka trzech zmiennych. Proces rozpoczęto z punktu $[0, 1, 1]$. Przyjęto $\gamma_{\text{prog}} = 0.55$

	EXTREM	EXTREM+ RBF approx.
x_1	1.000 003	1.002 752
x_2	1.000 006	1.003 494
x_3	1.000 012	1.008 847
g_{obj}	0.000 000	0.008 847
Liczba wart. funkcji	281	209+80
Średni błąd		0.000 953

Tabela 2.2. Porównanie znalezionego minimum oraz liczby wyznaczeń wartości funkcji celu dla algorytmu EXTREM oraz algorytmu EXTREM połączonego z aproksymacją funkcji celu za pomocą sieci RBF dla funkcji Rosenbrocka pięciu zmiennych. Punktem startowym optymalizacji był punkt $[0, 1, 1, 1, 1]$. Przyjęto $\gamma_{\text{prog}} = 0.65$

	EXTREM	EXTREM+ RBF approx.
x_1	0.999 998	0.998 981
x_2	0.999 996	0.998 029
x_3	0.997 990	0.996 059
x_4	0.999 985	0.992 051
x_5	0.999 971	0.984 196
g_{obj}	0.000 000	0.000 085
Liczba wart. funkcji	730	469+90
Średni błąd		0.000 029

który jest równoważny równaniu Laplace'a

$$\nabla(\nabla \times A_z) = 0 \tag{2.59}$$

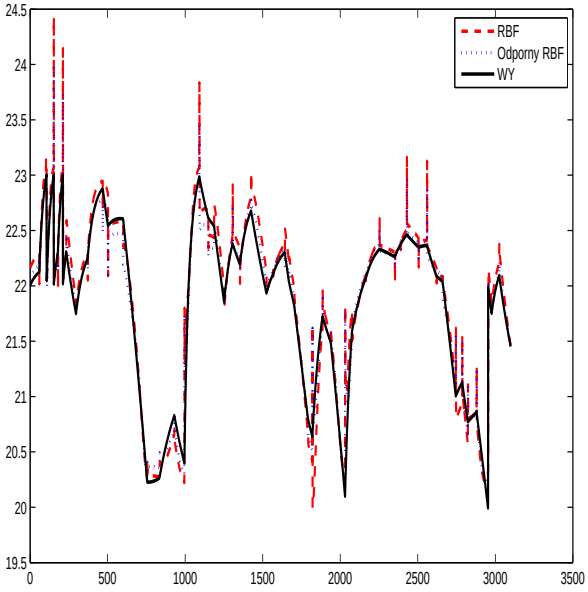
spełnianego przez składową z potencjału wektorowego A zdefiniowanego standardowo jako

$$\vec{B} = \nabla \times \vec{A}$$

w przestrzeni trójwymiarowej.

Pole magnetyczne jest obliczane z zależności $B_x = \frac{\partial A_z}{\partial y}$ oraz $B_y = -\frac{\partial A_z}{\partial x}$. Przechodząc do współrzędnych biegunowych (r, φ), równanie (2.59) przyjmuje postać

$$\frac{\partial^2 A_z}{\partial r^2} + \frac{1}{r}\frac{\partial A_z}{\partial r} + \frac{1}{r^2}\frac{\partial^2 A_z}{\partial \varphi^2} = 0.$$



Rysunek 2.15. Wartość funkcji celu dla algorytmu EXTREM oraz algorytmu EXTREM połączonego z aproksymacją funkcji celu za pomocą sieci RBF dla funkcji Rosenbrocka trzech zmiennych

Ustalając pole dla $r = 0$, otrzymujemy rozwiązanie

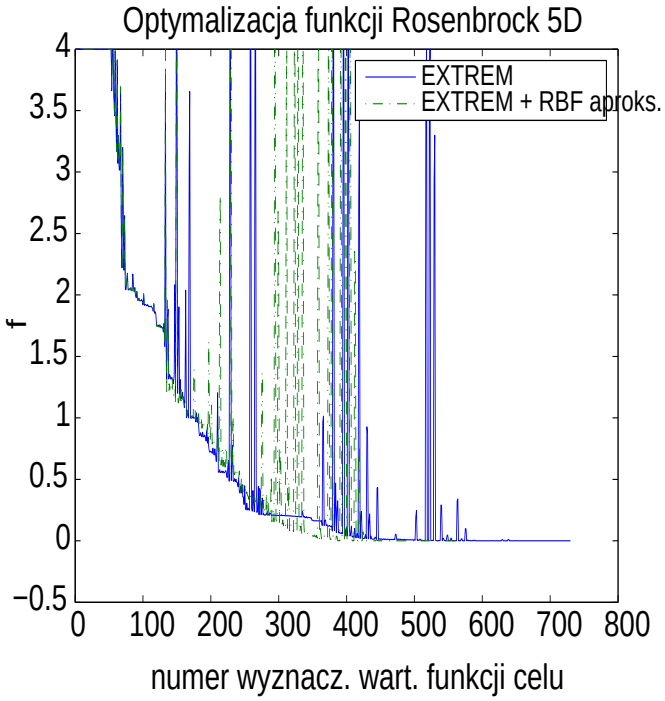
$$A_z(r,\varphi) = \sum_{n=1}^{\infty} r^n (G_n \cos(n\varphi) + H_n \sin(n\varphi)).$$

Radialna i kątowa składowa pola zapisują się wówczas jako szeregi

$$B_r(r,\varphi) = \frac{1}{r}\frac{\partial}{\partial\varphi} A_z(r,\varphi) = \sum_{n=1}^{\infty} r^{n-1}(B_n \sin(n\varphi) + A_n \cos(n\varphi)), \tag{2.60}$$

$$B_\varphi(r,\varphi) = -\frac{\partial}{\partial r} A_z(r,\varphi) = \sum_{n=1}^{\infty} r^{n-1}(B_n \cos(n\varphi) - A_n \sin(n\varphi)),$$

gdzie $B_n = -nG_n$ i $A_n = nH_n$. Jakość pola w aperturze jest formułowana za pomocą współczynników rozwinięcia składowej radialnej (2.60) na zadanym promieniu referencyjnym $r = r_{\text{ref}}$. Dla magnesów zderzacza cząstek elementarnych LHC $r_{\text{ref}} = 17$ mm.



Rysunek 2.16. Wartość funkcji celu dla algorytmu EXTREM oraz algorytmu EXTREM połączonego z aproksymacją funkcji celu za pomocą sieci RBF dla funkcji Rosenbrocka pięciu zmiennych

Aby zdefiniować funkcję celu, przekształćmy rozwinięcie (2.60) na okręgu o promieniu referencyjnym r_{ref} do postaci

$$B_r(r_{\text{ref}}, \varphi) = B_1(r_{\text{ref}}) \sum_{n=1}^{\infty} (b_n(r_{\text{ref}}) \sin n\varphi + a_n(r_{\text{ref}}) \cos n\varphi). \tag{2.61}$$

Ze względu na symetrię budowy magnesów zderzacza cząstek elementarnych LHC znikają z podanego rozwinięcia składniki $a_1(r_{\text{ref}}), a_2(r_{\text{ref}}), \ldots$. Celami w procesie optymalizacji zakrzywiających magnesów dipolowych (zob. [23]) dla zderzacza LHC są:

1. Pole magnetyczne o zadanym natężeniu strumienia $B_1 = 9.3T$.

2. Czystość pola magnetycznego na nominalnym poziomie natężenia prądu pobudzającego, tzn.

$$\begin{cases} \min b_2(r_{\text{ref}}), \\ \min b_3(r_{\text{ref}}), \\ \min b_4(r_{\text{ref}}), \\ \vdots \end{cases}$$

3. Czystość pola magnetycznego na poziomie prądu pobudzającego podczas wstrzyknięcia wiązki protonów do zderzacza cząstek elementarnych, tzn.

$$\begin{cases} \min b_{2,\text{inj}}(r_{\text{ref}}), \\ \min b_{3,\text{inj}}(r_{\text{ref}}), \\ \min b_{4,\text{inj}}(r_{\text{ref}}), \\ \vdots \end{cases}$$

4. Zmiana pola pomiędzy poziomem natężenia podczas wstrzyknięcia wiązki protonów do zderzacza cząstek elementarnych a poziomem nominalnym natężenia, tzn.

$$\begin{cases} \min \Delta b_2(r_{\text{ref}}) & = & \min(b_{2,\text{inj}}(r_{\text{ref}}) - b_2(r_{\text{ref}})), \\ \min \Delta b_3(r_{\text{ref}}) & = & \min(b_{3,\text{inj}}(r_{\text{ref}}) - b_3(r_{\text{ref}})), \\ \min \Delta b_4(r_{\text{ref}}) & = & \min(b_{4,\text{inj}}(r_{\text{ref}}) - b_4(r_{\text{ref}})), \\ \vdots \end{cases}$$

Tabela 2.3. Porównanie działania algorytmu EXTREM oraz metody SPELROA połączonej z algorytmem EXTREM do optymalizacji głównego magnesu dipolowego zderzacza cząstek elementarnych LHC. Proces minimalizacji został rozpoczęty z punktu $[88.0, 105.0, 79.0]$. Uzyskano przyspieszenie około 30%

	EXTREM	SPELROA
x_1	80.92	80.70
x_2	99.64	99.40
x_3	86.02	85.74
g_{obj}	-23.94	-24.18
Liczba obliczeń wart. g_{obj}	301	208
Liczba aproksymacji RBF g_{obj}		45

Tabela 2.4. Porównanie działania algorytmu EXTREM oraz metody SPELROA połączonej z algorytmem EXTREM do optymalizacji głównego magnesu dipolowego zderzacza cząstek elementarnych LHC. Proces minimalizacji został rozpoczęty z punktu $[95.0, 96.0, 85.0, 275.0, 8.0]$. Uzyskano przyspieszenie około 18.5%

	EXTREM	SPELROA
x_1	87.10	87.44
x_2	105.39	104.69
x_3	92.90	93.55
x_4	280.97	280.99
x_5	3.91	4.10
g_{obj}	15.51	15.66
Liczba obliczeń wart. g_{obj}	343	281
Liczba aproksymacji RBF g_{obj}		52

Funkcję celu zawierającą określone cele dla początkowych składowych b_n konstruuje się za pomocą metody ważenia celów, funkcji odległości oraz metod włączania ograniczeń (zob. [1], [23]). Zmiennymi, od których zależy optymalizowana funkcja celu, są parametry techniczne modelu magnesu – zob. rys. 2.17.

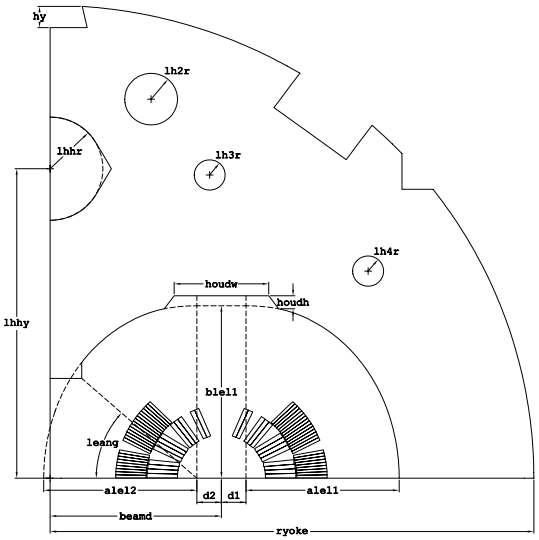
Wyniki numeryczne

Funkcja celu, jaką optymalizowaliśmy, skonstruowana została za pomocą metody ważenia celów i ma postać:

$$g_{\text{obj}}(\mathbf{x}) = \sum_{i=2}^{4} t_i \Delta b_i + \sum_{i=2}^{4} b_{i,\text{inj}} \quad \text{dla} \quad (t_2 = t_3 = 5, t_4 = 50), \tag{2.62}$$

gdzie b_2, b_3 i b_4 są współczynnikami kwadrupolowymi, sekstopolowymi oraz oktopolowymi odpowiednio. Parametrami, względem których prowadzona jest optymalizacja trzech zmiennych, są $\mathbf{x} = (x_1, x_2, x_3)$, gdzie $x_1 =$ alel1, $x_2 =$ blel1 oraz $x_3 =$ alel2 są parametrami kształtu kołnierza niemagnetycznego wokół cewki (rys. 2.17). Do optymalizacji użyliśmy algorytmu EXTREM [15] połączonego z metodą SPELROA. Proces minimalizacji rozpoczęto z punktu $[88.0, 105.0, 79.0]$. W tabeli 2.3 przedstawiamy porównanie znalezionych minimów funkcji celu oraz liczby wyznaczeń funkcji celu dla obydwu metod. Uzyskane za pomocą metody SPELROA przyspieszenie to około 30%.

W pracy [3] przedstawiliśmy również wyniki optymalizacji głównego magnesu dipolowego zderzacza cząstek elementarnych LHC dla funkcji (2.62) zależnej od



Rysunek 2.17. Parametryczna reprezentacja optymalizowanego jarzma magnesu LHC

pięciu parametrów projektu. Dodatkowo pod uwagę wzięto zmienne $x_4 = \mathsf{ryoke}$ oraz $x_5 = \mathsf{hy}$ (zob. rys. 2.17). Zastosowana metoda różniła się od prezentowanej w tej pracy jedynie sposobem określenia regionu wiarygodnego oraz sposobu wyboru parametru regularyzacji w konstrukcji aproksymatora funkcji celu. Różnica polegała na użyciu odległości Mahalanobisa zamiast wskaźnika $\gamma_{\mathbf{X}}$ oraz arbitralnego przedziału poszukiwania wartości parametru regularyzacji λ. Wyniki przedstawiono w tabeli 2.4.

2.5. Podsumowanie

W rozdziale tym omówiliśmy dwie metody zastosowania sieci neuronowych z radialnymi funkcjami bazowymi do przyspieszenia procesów optymalizacyjnych. Pierwsza metoda polega na minimalizacji modelu aproksymującego skonstruowanego w regionie wiarygodności, natomiast druga metoda polega na konstrukcji modelu aproksymującego dla punktu w regionie bogatym w dane i zastąpieniu bezpośredniego wyznaczania wartości funkcji jej aproksymacją. Dla drugiej metody przedstawiono wyniki numeryczne dla optymalizacji funkcji testowej, jeśli chodzi zarówno o błąd aproksymacji, jak i proporcję zmniejszenia liczby bezpośrednich wyznaczeń funkcji celu.

Zagadnieniem otwartym, nad którym pracujemy, jest ostrzejsze niż w twierdzeniu 15 oszacowanie błędu aproksymacji dla danego zbioru punktów danych.

Jednym z możliwych kierunków badań w celu określenia takiego oszacowania dla małych zbiorów danych jest możliwość wnioskowania o błędzie aproksymacji w punkcie, gdy wartość funkcji Lebesgue'a w danym punkcie jest mała, tzn. ~ 2, a założenia o gęstości zbioru punktów nie są spełnione.

Bibliografia

[1] Bazan M., *Wspomaganie procesów optymalizacyjnych za pomocą sieci neuronowych z radialnymi funkcjami aktywacji* (rozprawa doktorska), Instytut Informatyki, Automatyki i Robotyki, Wydział Elektroniki, Politechnika Wrocławska, 2010.

[2] Bazan M., *Search Procedure Exploiting Locally Regularized Objective Approximation. A Convergence Theorem for Direct Search Algorithms*, In Y. Tenne and C.-K. Goh, Computational Intelligence in Optimization. Adaptation Learning and Optimization, Vol. 7, 73–103, Springer, 2010.

[3] Bazan M., Russenschuck S., *Using neural networks to speed up optimization algorithms*, Eur. Phys. J., AP 12, 109–115, 2000.

[4] Bazan M., Aleksa M., Russenschuck S., *An improved method using radial basis function neural networks to speed up optimization algorithms*, IEEE Trans. on Magnetics, 38, 1081–1084, 2002.

[5] Bazan M., Aleksa M., Lucas J., Russenschuck S., Ramberger S., Völlinger C., *Integrated design of superconducting magnets with the CERN field computation program ROXIE,* Proc. 6$^{\text{th}}$ International Computational Accelarator Physics Conference, Darmstadt, Germany, September 2000.

[6] Bazan M., *Tikhonov regularization used to speed-up optimization algorithms*, ICCAM 2006, Leuven, Belgium, July 2006.

[7] Buhmann M.D., *Radial basis functions: Theory and Implementations*, Cambridge University Press, 2005.

[8] Conn A.R., Gold N.I.M., Toint Ph.L., *Trust region methods*, MPS-SIAM Series on Optimization, Philadelphia 2000.

[9] Conn A.R., Scheinberg K., Vincente L.N., *Geometry of interpolation sets in derivative free optimization*, Math. Programming: Series A and B, Vol. 111, Issue 1, 141-172, 2007.

[10] Conn A.R., Scheinberg K., Toint Ph.L., *On the convergence of derivative-free methods for unconstrained optimization*, w *Approximation Theory and Optimization: Tributes to M.J.D. Powell*, A. Iserles and M. Buhmann (eds.), 83–108, 1997.

[11] Davydov O., *Error Bound for Radial Basis Interpolation in Terms of a Growth Function*, Strathclyde Mathematics Research Report No. 24, 2006.

[12] Davydov O., Zeilfelder F., *Scattered data fitting by direct extension of local polynomials to bivariate splines*, Adv. Comp. Math., 21, 223–271, 2004.

[13] Goodman J.E., O'Rourke J., *Handbook of discrete and computational geometry*, Chapman & Hall/CRC, 2004.

[14] Hansen P.Ch., *Rank-Deficient and Discrete Ill-Posed Problems*, Numerical Aspects of Linear Inversion, SIAM, Philadelphia, 1998.

[15] Jacob H.G., *Rechnergestützte Optimierung statischer und dynamischer Systeme*, Springer, 1982.
[16] Jetter K., Stöckler J., Ward J., *Error estimates for scattered data interpolation on Spheres*, Math. Comput., 68, 733–747, 1999.
[17] Larsson E., Fornberg B., *Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions*, Comp. and Math. with Applications, 49, 103–130, 2005.
[18] Madych W.A., Nelson S.A., *Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation*, J. Approx. Theory, 70, 94–114, 1992.
[19] Micchelli Ch.A., *Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions,* Constructive Approximation, 2, 11–22, 1986.
[20] Norcowich F.J., Ward J.D., Wendland H., *Sobolev bounds on functions with scattered zeros, with applications to radial basis function surface fitting*, Math. Comput., 74, 643–763, 2005.
[21] Oeuvray R., *Trust region method based on radial basis functions with application on biomedical imaging*, Ecole Polytechnique Federale de Lausanne, 2005.
[22] Poggio T., Girosi F., *A theory of networks for approximation and learning*, A.I. Memo No. 1140, C.B.I.P. Paper No. 31, July 1989.
[23] Russenschuck S., *ROXIE: Routine for the optimization of magnet X-section, inverse field calculation and coil end design*, CERN Yellow Report 99–01, April 1999.
[24] Schaback R., *Error estimates and condition number for radial basis function interpolation*, *Adv. Comput. Math.*, 3, 251–264, 1995.
[25] Schaback R., *Improved error bounds for scattered data interpolation by radial basis functions*, Mathematics of Computation, 68, 201–216, 1999.
[26] Schaback R., *Multivariate Interpolation by Polynomials and Radial Basis Functions*, Constr. Approx., 21, 293–317, 2005.
[27] Schaback R., *Native Hilbert Spaces for Radial Basis Functions I*, The new development in Approximation Theory, Birkhäuser, Basel, 1999.
[28] Tikhonov A.N., Arsenin V.A., *Solution of ill-posed problems*, Winston & Sons, Washington, 1977.
[29] Wahba G., *Spline models for obsevational data*, SIAM, Philadelphia, 1990.
[30] Wendland H., *Scattered Data Approximation*, Cambridge University Press, 2005.
[31] Wendland H., Rieger C., *Approximate Interpolation with Applications to Selecting Smoothing Parameters*, *Numerische Mathematik*, 101, 643–662, 2005.
[32] Wendland H., *Gaussian Interpolation Revisited*, In K. Kopotun, T. Lyche, M. Neamtu, eds., Trends in Approximation Theory, 427–436. Nashville, Vanderbilt University Press, 2001.
[33] Wendland H., *Computational aspects of radial basis function approximation*, w K. Jetter *et al.* (eds.) *Topics in Multivariate Approximation and Interpolation*, Elsevier B.V., 231–256, 2005.
[34] Yun-Wei S., Yu-Huang Q., *A note on the extended Rosenbrock function*, Evolutionary Computation, 14, No. 1, 119–126, 2006.

Rozdział 3

Wykrywanie małych zmian w układach chaotycznych

Mateusz Tykierko

> Nauka nie stara się wyjaśnić, a nawet niemal nie stara się interpretować, zajmuje się ona głównie budowaniem modeli. Model rozumiany jako matematyczny twór, który po dodaniu słownej interpretacji, opisuje obserwowane zjawiska. Jedynym i właściwym uzasadnieniem takiego tworu matematycznego jest oczekiwanie, że sprawdzi się on w działaniu.
>
> *J. von Neumann*

3.1. Wprowadzenie

Wykrywanie zmian, nowości lub defektów jest ważnym problemem pojawiającym się w wielu zagadnieniach związanych z kontrolą jakości, diagnostyką, robotyką, służbą zdrowia, ochroną środowiska czy sieciami komputerowymi.

Wykryta zmiana traktowana jest jako bodziec lub ostrzeżenie umożliwiające: uniknięcie katastrofy przemysłowej, zredukowanie czasu przestoju produkcji, poprawienie jakości produktu, ratowanie życia, modyfikacje sposobu leczenia, a także metod uprawy roślin, analizę kryptograficzną itd. Można ją również rozpatrywać jako element prediagnostyczny, który (bez matematycznego opisu zjawiska) ma odpowiedzieć na pytanie czy w układzie nastąpiła, czy też nie nastąpiła zmiana.

Większość obserwowanych procesów naturalnych czy systemów stworzonych ręką ludzką ma nieliniową i niejednokrotnie chaotyczną naturę. W porównaniu do dobrze zbadanego zagadnienia, dotyczącego wykrywania zmian w układach liniowych, metod rozważających nieliniowość jest niewiele.

Niniejszy rozdział jest podzielony na dwa główne podrozdziały. Obejmują one zagadnienia związane z modelowaniem chaotycznych strumieni danych za pomocą sieci RBF oraz wykrywaniem zmian w tych strumieniach.

3.2. Modelowanie układu chaotycznego

Problem modelowania dynamiki, na podstawie danych eksperymentalnych polega na poszukiwaniu aproksymacji wielowymiarowej przestrzeni reprezentującej odwzorowanie zmieniające stan systemu. Modele układów dynamicznych tworzone są na podstawie zebranych informacji o badanym obiekcie. Ilość dostępnych informacji określa możliwe techniki modelowania. W rozdziale rozważane jest zadanie modelowania układów chaotycznych za pomocą sztucznych sieci neuronowych RBF (ang. *Radial Basis Function*). Zakładany jest brak wiedzy o wewnętrznej strukturze obiektu ani jego własnościach fizycznych – w szczególności nie jest zakładany rzędu układu. Brak tych informacji kompensowany jest większymi wymaganiami co do wielkości zbioru dostępnych danych.

Zakłada się jedynie determinizm i stacjonarność modelowanego układu. Jedyna dostępna informacja ograniczona jest do jednowymiarowego szeregu czasowego powstałego z próbkowania wyjść układu. Podane ograniczenia umożliwiają analizę rzeczywistych układów dynamicznych, o których informacje uzyskujemy na podstawie pomiarów eksperymentalnych.

Sieci o radialnych funkcjach bazowych RBF są dobrze znanym, uniwersalnym narzędziem wielowymiarowej aproksymacji dowolnych nieliniowych funkcji ciągłych [11], [13], [47]. Ten typ sieci jest więc znakomitym narzędziem modelowania zjawisk nieliniowych, jakimi są zachowania układów chaotycznych.

Modelowanie układów chaotycznych za pomocą sieci RBF można znaleźć w następujących pozycjach literaturowych [12], [24], [45]. Podstawową różnicą między tymi pracami, a dalej opisanym postępowaniem jest użycie algorytmu konstrukcyjnego oraz brak reestymacji wartości regularyzacji.

Proponowany jest następujący algorytm:

Algorytm 1. *Modelowanie układów chaotycznych za pomocą sieci RBF:*

1. *Dla deterministycznego jednowymiarowego nieliniowego szeregu pomiarów wyjścia układu dynamicznego* $Y = [y(1), y(2), \ldots, y(N)]$ *zrekonstruuj przestrzeń stanów.*
2. *Oblicz niezmienniki dynamiki, tj. wymiar fraktalny* d_E *i maksymalny wykładnik Lapunowa* λ_L[1].
3. *Ze zrekonstruowanego szeregu utwórz zbiór uczący* $\mathbf{X} = [\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(p)]$ *dla sieci neuronowej według procedury opisanej w podrozdz. 3.2.6 o liczbie elementów* $p < N$.
4. *Korzystając z algorytmu konstrukcyjnego selekcji postępującej (ang. forward selection) zbuduj model oparty na sieci RBF (algorytm 3).*

[1] Z powodu niewystarczającej dokładności algorytmów estymacji wykładników Lapunowa nie jest zalecane obliczanie spektrum wykładników.

5. *Zainicjuj model p-tym elementem ciągu uczącego i rekurencyjnie predykuj kolejne N wartości wyjścia modelu.*
6. *Porównaj predykowane wyjście $\hat{X}$ z rzeczywistymi wartościami. Sprawdź czy czas objęty przez kolejne poprawnie predykowane wartości $x \approx \hat{x}$ jest zgodny z horyzontem predykcji.*
7. *Oblicz na predykowanym szeregu wartość wymiaru fraktalnego oraz maksymalnego wykładnika Lapunowa. Porównaj je z wartościami tych niezmienników obliczonymi w punkcie 2. Jeżeli występuje różnica większa niż dokładność użytej metody estymacji, przejdź do kroku 1.*

Szczegółowy opis teorii rekonstrukcji, metod doboru parametrów zanurzenia oraz obliczania niezmienników dynamiki czytelnik może znaleźć w pracach [2], [9], [18], [24], [29], [41], [49], [50], [53].

Metody obliczania niezmienników dynamiki, jakimi są wymiar fraktalny czy wykładnik Lapunowa, na podstawie szeregów czasowych czytelnik może znaleźć w pracach [3], [8], [20], [31], [48], [50], [52], [54]. Natomiast metody obliczeniowe na danych strumieniowych opisane są w artykule [55].

O modelu układu chaotycznego możemy mówić jeżeli generowane przez niego wartości spełniają następujące warunki:

– zachowanie krótkookresowe – wartości generowane przez model powinny, przez czas wyznaczony horyzontem predykcji, być zbliżone do wartości wyjścia układu,
– zachowanie długookresowe – wartości niezmienników dynamiki, tj. wymiar fraktalny, dodatnie wykładniki Lapunowa są zbliżone do wartości niezmienników modelowanego układu.

Podstawowym założeniem proponowanego algorytmu jest zapewnienie poprawnej predykcji kolejnych wartości układu jak również zachowanie wartości jego niezmienników.

Do określenia horyzontu predykcji dla modeli układów chaotycznych stosuje się wykładnik Lapunowa. Jest on wielkością określającą średnią rozbieżność dwóch trajektorii startujących z bliskich wartości początkowych. W ujęciu modelowania wykładnik jest używany do określenia poprawności modelu oraz horyzontu czasowego, dla którego predykcja jest możliwa [1], [3], [4]. Po przekroczeniu tego czasu dokładność predykcji drastycznie się załamuje. Związek pomiędzy maksymalnym czasem predykcji $t_{\max}$ a wykładnikiem Lapunowa opisany jest zależnością [17]:

$$t_{\max} \approx \frac{\ln(N)}{\lambda_L d_E} \tag{3.1}$$

gdzie d_E jest wymiarem fraktalnym układu, N liczbą dostępnych danych, a λ_L maksymalnym wykładnikiem Lapunowa. Z zależności tej wynika, że bez względu

na dokładność modelu i liczbę dostępnych danych dla układów chaotycznych nie można poprawnie przewidzieć więcej wartości układu niż kilka wielokrotności maksymalnego wykładnika Lapunowa $1/\lambda_L$.

3.2.1. Sieć radialna — RBF

Sieci radialne RBF są jednokierunkowymi sieciami neuronowymi składającymi się z warstwy wejściowej, ukrytej i wyjściowej. Ich struktura została wprowadzona przez Brommohead i Lowe w pracy [7]. Literatura obfituje w artykuły dotyczące teorii i zastosowania tych sieci [6], [10], [12], [15], [22], [23], [34], [39], [40], [45], [44].

Sieci RBF są powiązane z estymatorem Parzena gęstości prawdopodobieństwa oraz estymatorem Nadaray'a-Watsona [22], [23], [56].

Modelowanie sygnałów chaotycznych za pomocą sieci o bazach radialnych rozważane jest w pracach [12], [24], [25], [27], [32], [45].

Wyjście sieci RBF składającej się z m neuronów warstwy ukrytej dla wektora wejściowego $\mathbf{x}$ o wymiarze l jest odwzorowaniem $R^l \rightarrow R$ najczęściej opisanym równaniem:

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x}) + w_0, \tag{3.2}$$

gdzie w_i są wagami sieci, natomiast $h_i(\mathbf{x})$ są funkcjami radialnymi.

Zbiór parametrów całkowicie opisujący sieć RBF składa się z:
- liczby wejść – wymiar l wektora $\mathbf{x}$,
- liczby m neuronów warstwy ukrytej,
- parametrów funkcji radialnej – położenie centrum c oraz szerokość funkcji σ,
- wartości wag w_i,
- wartości parametru regularyzacji λ.

Parametr regularyzacji λ nie jest parametrem obowiązkowym, jednakże według [24], stosując sieć RBF do modelowaniu układów chaotycznych, jest on niezbędny. Konieczność zastosowania regularyzacji potwierdzają również eksperymenty przeprowadzone przez autora pracy. Brak regularyzacji uniemożliwiał zbudowanie poprawnego modelu.

Algorytm 2. *Dobór parametrów modelu:*
1. *Zbuduj zbiór uczący* $\{(\mathbf{x}_i, y_i)\}_{i=1}^{p}$ *o* p *elementach, gdzie wymiar* $\mathbf{x}_i$ *jest równy* l *i określa liczbę wejść (podrozdz. 3.2.6).*
2. *Wybierz parametry funkcji radialnej, typ funkcji,* $\mathbf{c}$ *– centra oraz* σ *– szerokość (podrozdz. 3.2.5).*
3. *Wybierz liczbę neuronów* m *warstwy ukrytej (rozdział 3.2.7), wartość parametru regularyzacji* λ*(podrozdz. 3.2.4),*

4. *Dobierz wagi w_i, które minimalizują błąd pomiędzy zbiorem uczącym a wyjściem z sieci (podrozdz. 3.2.8).*

3.2.2. Terminy pomocnicze

Podrozdział ten zawiera opis elementów sieci RBF niezbędnych do obliczania wag i błędu sieci. Funkcja wyjścia sieci (3.2) $f(\mathbf{x}_i)$ w zapisie macierzowym przyjmuje postać:

$$\hat{y}(\mathbf{x}_i) = \sum_{j=1}^{m} w_j \phi_j(\mathbf{x}_i) = \phi_{\mathbf{i}}^{\mathbf{T}} \mathbf{w} \tag{3.3}$$

gdzie $\mathbf{w}$ jest wektorem wag, $\phi_{\mathbf{i}}^{\mathbf{T}} = [h_1(\mathbf{x}_i)\phi_2(\mathbf{x}_i) \cdots \phi_m(\mathbf{x}_i)]$
jest i-tym wierszem macierzy Greena :

$$\mathbf{G} = \begin{bmatrix} \phi_1(\mathbf{x_1}) & \phi_2(\mathbf{x_1}) & \cdots & \phi_m(\mathbf{x_1}) \\ \phi_1(\mathbf{x_2}) & \phi_2(\mathbf{x_2}) & \cdots & \phi_m(\mathbf{x_2}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x_p}) & \phi_2(\mathbf{x_p}) & \cdots & \phi_m(\mathbf{x_p}) \end{bmatrix}, \tag{3.4}$$

m jest liczbą neuronów warstwy ukrytej, p – liczbą elementów zbioru uczącego $\{\mathbf{x}_i, y_i\}_{i=1}^{p}$. Wartość $\phi_j(\mathbf{x}_i)$ jest odpowiedzią j-tej funkcji bazowej i-tego przykładu zbioru uczącego.

Odpowiedź sieci na wszystkie p elementów macierzy G, zbudowanej z ciągu uczącego, przyjmuje postać:

$$\hat{\mathbf{y}} = \begin{bmatrix} \phi_1^T \\ \phi_2^T \\ \vdots \\ \phi_p^T \end{bmatrix} \mathbf{w} = \mathbf{G}\mathbf{w}, \tag{3.5}$$

przy czym indeks górny ^{T} oznacza macierz transponowaną.

Optymalny wektor wag $\mathbf{w}$ obliczany jest przez minimalizację funkcji błędu:

$$E = \sum_{i=1}^{p} (y_i - \hat{y}(\mathbf{x}_i))^2 + \lambda \sum_{j=1}^{m} w_j^2, \tag{3.6}$$

gdzie λ jest parametrem regularyzacji stosowanym w celu uniknięcia zbytniego dopasowania do zbioru uczącego. Szczegółowy opis i metody doboru parametru regularyzacji znajduje się w podrozdziale 3.2.4.

W celu znalezienia minimum funkcji błędu równanie (3.6) jest różniczkowane:

$$\frac{\partial E}{\partial w_j} = 2 \sum_{i=1}^{p} (\hat{y}(\mathbf{x}_i) - y_i) \frac{\partial f(\mathbf{x}_i)}{\partial w_j} + 2\lambda w_j. \tag{3.7}$$

Podstawiając za pochodną cząstkową $\frac{\partial f(\mathbf{x}_i)}{\partial w_j} = \phi_j(\mathbf{x}_i)$ i przyrównując wynik do zera otrzymujemy:

$$\sum_{i=1}^{p} \hat{y}(\mathbf{x}_i)\phi_j(\mathbf{x}_i) + \lambda w_j = \sum_{i=1}^{p} y_i \phi_j(\mathbf{x}_i), \tag{3.8}$$

gdzie ϕ_j jest kolumną macierzy (3.4). Korzystając ze wzoru (3.5), powyższe równanie przyjmuje następującą postać macierzową:

$$\mathbf{G}^T\hat{\mathbf{y}} + \lambda\mathbf{w} = \mathbf{G}^T\mathbf{y},$$
$$\mathbf{G}^T\mathbf{G}\mathbf{w} + \lambda\mathbf{w} = \mathbf{G}^T\mathbf{y}.$$

Rozwiązanie przyjmuje postać

$$\mathbf{w} = (\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{G}^T y. \tag{3.9}$$

Stosując wzór (3.9) błąd między wyjściem sieci a zbiorem uczącym wyraża się równaniem:

$$\mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{G}\mathbf{w} = \mathbf{y} - \mathbf{G}((\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I}_m)^{-1}\mathbf{y}) = \mathbf{P}\mathbf{y}, \tag{3.10}$$

gdzie

$$\mathbf{P} = \mathbf{I}_p - \mathbf{G}(\mathbf{G}^{\mathbf{T}}\mathbf{G} + \lambda\mathbf{I}_{\mathbf{m}})^{-1}\mathbf{G}^T \tag{3.11}$$

jest macierzą rzutowania. I_p oraz I_m są macierzami jednostkowymi o stopniu p i m.

Macierz rzutowania jest macierzą kwadratową, która ortogonalnie rzutuje wektor przestrzeni p-wymiarowej na m-wymiarową podprzestrzeń definiowaną przez model.

Optymalny model, w sensie metody najmniejszych kwadratów, charakteryzuje się minimalną różnicą między zbiorem uczącym a wyjściem sieci. Różnica ta jest iloczynem macierzy rzutowania i zbioru uczącego. Sumaryczny błąd sieci wyrażony z użyciem macierzy rzutowania ma postać:

$$E = \sum_{i=1}^{p}(y_i - \hat{y}(\mathbf{x}_i))^2 = (\mathbf{y} - \hat{\mathbf{y}})^T(\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{y}^T\mathbf{P}^T\mathbf{P}\mathbf{y} = \mathbf{y}^T\mathbf{P}^2\mathbf{y}. \tag{3.12}$$

3.2.3. Kryterium doboru modelu

Kryterium doboru modelu jest estymatą, która umożliwia ocenę dokładności jakości modelu, przez co ułatwia wybór właściwego modelu ze zbioru potencjalnych modeli. Za najlepszy model uważany jest ten, dla którego wartość

kryterium jest najmniejsza. Do podstawowych kryteriów wyboru modelu należą: kryterium walidacji krzyżowej (ang. *Cross-Validation*), kryterium globalnej walidacji krzyżowej GCV (ang. *Global Cross-validation*), bayesowskie kryterium informacyjne BIC (ang. *Bayesian Information Criteria*), kryterium informacyjne Akaike AIC (ang. *Akaike Information Criteria*), kryterium nieobciążonej oceny wariancji UEV (ang. *Unbiased Estimate of Variance*). Przegląd kryteriów doboru modelu dla nieliniowych szeregów czasowych opisano w pracy [33]. Wszystkie wymienione kryteria bazują na wartości błędu SSE (3.12) obliczonego na zbiorze uczącym oraz liczbie parametrów modelu.

Wybrano kryterium GCV opisane wzorem [38]:

$$\sigma^2_{GCV} = \frac{pE}{(trace(\mathbf{P}))^2} \tag{3.13}$$

gdzie $trace(\mathbf{P})$ jest śladem macierzy rzutowania $\mathbf{P}$.

W przypadku sieci RBF kryterium doboru modelu służy do wyboru liczby neuronów warstwy ukrytej.

Wybór GCV do oceny tworzonych modeli spowodowany był chęcią porównania i sprawdzenia wyników przedstawionych w pracy [12]. Autor cytowanej pracy twierdzi, że użycie kryterium BIC ułatwia otrzymanie lepszych rezultatów w porównaniu z GCV.

3.2.4. Parametr regularyzacji

Głównym celem stosowania regularyzacji jest uniknięcie zbytniego dopasowania sieci do zbioru uczącego. Została ona wprowadzona do sieci RBF w pracy [13]. Przegląd metod korzystających z regularyzacji można znaleźć w pracy [23].

Parametr regularyzacji λ nie jest obowiązkowy, jednak według [24] podczas modelowania układów chaotycznych jest niezbędny.

W pracach [12], [45] dotyczących modelowania układów chaotycznych stosowana jest reestymacja wartości regularyzacji z użyciem kryteriów BIC i GVC podczas uczenia sieci. Natomiast w proponowanej przez autora procedurze wartość parametru regularyzacji jest ustalona w krokach wstępnych algorytmu i nie podlega zmianie w trakcie budowania modelu. Upraszcza to znacznie procedurę obliczeniową. Przeprowadzone eksperymenty wykazały, iż stała wartość regularyzacji nie pogarsza jakości modelu.

3.2.5. Radialne funkcje bazowe i ich szerokość

Radialna funkcja bazowa umieszczana jest w neuronach warstwy ukrytej. Odpowiedź funkcji radialnych zależy od odległości argumentu funkcji od jej centrum.

Typ funkcji bazowej określa czy jej wartość będzie rosła czy malała. Wartość odpowiedzi funkcji gaussowskiej maleje wraz ze wzrostem odległości od centrum, natomiast wartość funkcji multikwadratowej rośnie. Innymi słowy, odpowiedź może być lokalna, gdy wartości funkcji są większe w pobliżu centrum (funkcja Gaussa) lub globalna (funkcja mulitkwadratowa), gdy jej wartość rośnie do nieskończoności wraz z oddalaniem się od centrum. Przegląd funkcji radialnych i ich własności znajduje się w pracy [26].

W dalszych rozważaniach będzie rozpatrywana sieć o bazach gaussowskich. Dla jednowymiarowego wektora wejściowego $\mathbf{x}$ (wejście skalarne) i dla funkcji Gaussa bazowa funkcja radialna przyjmuje postać

$$\phi(x) = e^{-\frac{(x-c)^2}{\sigma^2}}, \tag{3.14}$$

gdzie σ jest parametrem określającym szerokość (gładkość) funkcji radialnej.

Można wyróżnić dwie podstawowe klasy metod wyznaczania szerokości funkcji radialnych. Pierwsza klasa wiąże się z ustaleniem równej szerokości σ dla wszystkich funkcji. W pracy [22] autor zaproponował następującą formułę:

$$\sigma = \frac{d_{\max}}{\sqrt{2m}}, \tag{3.15}$$

gdzie $d_{\max}$ jest maksymalną odległością pomiędzy dwoma dowolnymi centrami. Taki wybór szerokości gwarantuje, że gaussowska funkcja radialna nie będzie zbyt gładka lub ostra. Gdyby dane były rozłożone równomiernie w przestrzeni wejściowej, wówczas powyższy wzór dawałby optymalne wartości σ.

Drugą klasą metod charakteryzuje indywidualne ustalanie szerokości dla każdej funkcji bazowej. Jedną z metod jest obliczanie odchylenia standardowego odległości pomiędzy centrum a danymi. W pracy [35] autorzy zaproponowali użycie metody najbliższych sąsiadów. Szerokość j-tej funkcji bazowej opisana jest wzorem:

$$\sigma_j = \frac{1}{r}\left(\sum_{i=1}^{r} \|\mathbf{c}_i - \mathbf{c}_j\|^2\right)^{1/2}, \tag{3.16}$$

gdzie $\mathbf{c}_i$ są najbliższymi sąsiadami funkcji o centrum w $\mathbf{c}_j$. Ta klasa metod korzysta z informacji o rozłożeniu danych w przestrzeni wejściowej. W pracy [5] autorzy zaproponowali połączenie obu danych klas. Takie podejście nie jest jednak realizowalne w przypadku algorytmu konstrukcyjnego stosowanego w rozdziale ze względu na zmienną liczbę neuronów.

3.2.6. Zbiór uczący

Zbiór uczący tworzony jest z dostępnych wejść i wyjść obiektu oraz apriorycznej wiedzy, którą mamy o modelowanym obiekcie.

W pracy rozważane są układy autonomiczne. Informacja o układzie ograniczona jest do jednowymiarowego szeregu czasowego, składającego się z wartości wyjścia układu mierzonego w równoodległych momentach czasu. Modelowany obiekt traktowany jest jako czarna skrzynka, o której strukturze niczego nie wiemy.

Do modelowania układów nieliniowych powszechnie stosowane są reprezentacje typu autoregresyjnego (NARX). Takie podejście wymaga określenia rzędu oraz opóźnienia. Dla sieci neuronowej parametry te definiują strukturę wejściową sieci.

Tworzenie zbioru uczącego wprost z jednowymiarowego szeregu czasowego nie umożliwia w ogólnym przypadku poprawnego modelowania układu. Standardowym rozwiązaniem analizy szeregów czasowych jest okno czasowe.

Szereg czasowy składa się z pomiarów wyjścia układu dynamicznego próbkowanych w równych odstępach czasu. Wektor wejściowy sieci powstaje z aktualnej wartości szeregu czasowego i jego wcześniejszych wartości. Zachodzi pytanie, jakiej liczby regresorów należy użyć oraz które z wcześniejszych wartości szeregu mają być wykorzystane? Innymi słowy, jaki wymiar n ma mieć wektor wejściowy i jaka ma być wartość opóźnienia τ między wyrazami oryginalnego szeregu?

Znanych jest wiele podejść do zagadnienia doboru regresorów w sieciach neuronowych. Jedną z częściej stosowanych procedur jest porównywanie modeli zbudowanych różnymi wartościami parametrów. Kryteriami porównania są jakość predykcji, błąd sieci, wielkość struktury itp. Takie podejście może prowadzić do błędnego wyboru wartości parametrów. Inną stosowaną metodą jest budowanie dużej sieci neuronowej, a następnie usuwanie zbędnych elementów na podstawie zastosowanych kryteriów.

Podane metody wyboru są czasochłonne i cechują się dużą złożonością obliczeniową. Alternatywą są heurystyki niewymagające każdorazowego uczenia sieci w celu określenia parametrów struktury wejściowej.

Proponowana metoda bazuje na mechanizmach rekonstrukcji dynamiki, które zapewniają wyznaczenie parametrów regresji. Ważną cechą takiego podejścia jest zachowanie informacji o obiekcie. Zbiór wejściowy reprezentuje zrekonstruowaną przestrzeń stanu modelowanego układu.

Szczegółowy opis teorii rekonstrukcji, metod doboru parametrów zanurzenia oraz obliczania niezmienników dynamiki czytelnik może znaleźć w pracach [2], [9], [18], [24], [29], [41], [49], [53].

Warunkiem wystarczającym do poprawnej rekonstrukcji jest $n \geqslant 2d_E$, gdzie d_E jest wymiarem fraktalnym rozważanego układu dynamicznego. Do wyznaczania opóźnienia τ i wymiaru n zastosowano metodę wzajemnych informacji [18] i fałszywych sąsiadów [29].

Znając wymiar i opóźnienie, wektor wejściowy można tworzyć dwojako: używając opóźnienia wprost, co prowadzi do następującej postaci wektora:

$$\mathbf{x}(t) = [x(t), x(t-\tau), ..., x(t-(n-1)\tau)] \tag{3.17}$$

lub stosując opóźnienie jako ograniczenie liczby elementów. W tym przypadku wektor będzie miał postać:

$$\mathbf{x}(t) = [x(t), x(t-1), ..., x(t-(n-1)\tau)]. \tag{3.18}$$

Innymi słowy, wektor wejściowy tworzony jest z oryginalnego szeregu czasowego przez próbkowanie w momentach czasu wyznaczonych opóźnieniem i czasem określonym przez wymiar zanurzenia.

Ostatecznie element zbioru uczącego przyjmie postać:

$$\{[x(t), x(t-\tau), ..., x(t-(n-1)\tau)], x(t+1)\}. \tag{3.19}$$

3.2.7. Dobór liczby neuronów

Zasadniczym problemem w projektowaniu sieci neuronowych jest wyznaczenie liczby neuronów warstwy ukrytej. Zbyt mała ich liczba powoduje niedostateczne dopasowanie do zbioru uczącego, natomiast zbyt duża nadmierne dopasowanie.

Liczbę neuronów można ustalić a priori [36] lub wyznaczyć na podstawie algorytmu konstrukcyjnego. Sposób realizacji na podstawie algorytmu konstrukcyjnego polega na budowaniu struktury sieci przez dodawanie lub odejmowanie elementów warstwy ukrytej w poszczególnych krokach algorytmu. Innymi słowy, struktura sieci zmienia się „dynamicznie" w zależności od przyjętego kryterium.

W przypadku gdy struktura sieci jest ustalona z góry, w procesie uczenia przeszukiwana jest przestrzeń parametrów zdefiniowana przez stałą liczbę neuronów. Każdorazowe dodanie lub usunięcie neuronu warstwy ukrytej powoduje zmianę wymiaru przeszukiwanej przestrzeni. Należy zauważyć, iż zbyt mała liczba neuronów uniemożliwia dostateczne zredukowanie błędu dopasowania na zbiorze danych uczących, natomiast zbyt duża ich liczba powoduje wzrost błędu uogólniania na zbiorze testującym.

Ekstremalnym podejściem jest budowanie sieci składającej się z neuronów umieszczonych w punktach wszystkich elementów zbioru uczącego[2]. Struktura takiej sieci jest bardzo duża bądź nieskończona w przypadku strumieni danych, lub układów o niepowtarzających się wartościach. Charakteryzuje się również małą zdolnością do uogólnienia.

[2] Taka konstrukcja sieci o odpowiednio małej szerokości funkcji poprawnie klasyfikuje zbiór uczący bez względu na jego skomplikowanie.

Pierwsze rozważania dotyczące dynamicznego tworzenia struktury sieci można znaleźć w pracy [42]. Autor zaproponował metodę sieci przydziału zasobów RAN (ang. *Resource Allocation Network*). Polega ona na dodawaniu nowego neuronu w chwili, gdy odległość między istniejącymi centrami a nowym elementem ze zbioru uczącego jest znacząca lub nowy element powoduje wzrost błędu powyżej zadanego progu. W pracy [19] został zaproponowany algorytm wzrastających struktur komórkowych GCS (ang. *Growing Cell Structures*), który w przeciwieństwie do RAN nie jest wrażliwy na zaszumiony ciąg uczący. Metoda aproksymuje miejsce dodania nowego neuronu pomiędzy neuronem wprowadzającym największy błąd a jego sąsiadem z przestrzeni wejściowej. Ze względu na dobór pozycji centrów, GCS można rozpatrywać jako rozszerzenie sieci SOM (ang. *Self Organization Maps*). Algorytm ten daje również możliwość usuwania nadmiarowych neuronów, których centra pokrywają się i mają podobne wartości wag. W pracy [14] autorzy wskazują na dwie podstawowe wady GCS. Pierwsza z nich związana jest z brakiem stabilności w strukturze sieci powodowanym ciągłym dodawaniem i usuwaniem neuronów. Druga zaś wada GCS leży w algorytmie ustalania centrów, który powoduje dryft funkcji radialnych w kierunku najnowszych danych wejściowych. Skutkuje to łatwym zapominaniem sieci. Autorzy proponują metodę modyfikacji GCS nazwaną DCL (ang. *Dynamic Competitive Learning*), która niweluje pierwszą wadę GCS oraz stosują dynamiczne drzewa regresyjne (ang. *Dynamic Regression Tree*) do usunięcia drugiej. Podobnie autor pracy [40] stosuje drzewa regresyjne do wyznaczania liczby neuronów warstwy ukrytej i ich parametrów.

Modyfikacje podejścia RAN z użyciem rozszerzonych filtrów Kalmana, zamiast podejścia gradientowego, można znaleźć w pracach [37], [57].

Odmienny sposób proponują autorzy pracy [28], stosując metodę usuwania zbędnych połączeń (ang. *prunning*) do znalezienia małej struktury sieci. Podobne podejście, selekcja wsteczna (ang. *backward selection*), opisane jest w pracy [38]. W pracy [16] można znaleźć przegląd metod wyboru podzbiorów i ich porównanie.

W niniejszej pracy do realizacji zadania modelowania układów chaotycznych została użyta metoda selekcji postępowej (ang. *forward selection*). Jej pierwsza implementacja znajduje się w pracy [15], wersję z regularyzacją opisano natomiast w pracy [38]. Opis implementacji znajduje się w następnym podrozdziale.

3.2.8. Metoda selekcji postępowej

Odnalezienie optymalnego zbioru o rozmiarze m w $2^p - 1$ możliwych zbiorach metodą przeglądu zupełnego jest w przypadku ogólnym niepraktyczne. Niezbędne jest więc użycie heurystyki, która wyłoni dobry (choć nieoptymalny) podzbiór. Jedną z takich heurystyk jest selekcja postępowa.

Algorytm 3. *Budowanie sieci RBF metodą selekcji postępowej:*

1. *Z początkowych p elementów odtworzonego zbioru wektorów przestrzeni stanów* $\mathbf{X}$ *zbuduj zbiór uczący. Elementy zbioru przyjmują postać*

$$\{[x(t), x(t-\tau), ..., x(t-(n-1)\tau)], x(t+1)\}. \tag{3.20}$$

2. *Wybierz funkcję radialną.*
3. *Utwórz macierz potencjalnych neuronów* $\mathbf{G}_{pxp}$*, umieszczając centra w punktach zbioru uczącego.*
4. *Oblicz szerokość centrów* σ *według wzoru:*

$$\sigma = \frac{\max(\text{dis}(x(i), x(j))}{\sqrt{2m}}; i, j = 1, 2, \ldots, p, \tag{3.21}$$

   gdzie dis *jest odległością pomiędzy centrami* i, j*.*
5. *Dodaj do struktury sieci kolumnę* ϕ_j *macierzy* $\mathbf{G}$*, której dodanie minimalizuje aktualny błąd sieci.*
6. *Jeżeli nie jest spełniony żaden z warunków stopu, przejdź do kroku 5, w przeciwnym razie STOP.*

Rozpoczynając od pustego zbioru, w każdej iteracji algorytmu dodawana jest pojedyncza funkcja bazowa ze zbioru kandydatów, będącego macierzą (3.4) zawierającą wszystkie elementy ciągu uczącego. Wybierany jest neuron, który najbardziej redukuje błąd opisany równaniem (3.6). Proces dodawania kolejnych neuronów jest zatrzymywany, jeżeli zostanie spełniony którykolwiek z warunków stopu. Innymi słowy, metoda poszukuje podzbioru funkcji bazowych o minimalnym błędzie predykcji w dyskretnej przestrzeni podzbiorów zbioru neuronów o ustalonych centrach i szerokościach.

Zbiór kandydatów tworzony jest z całego zbioru uczącego. Wybór dokonywany jest na symetrycznej macierzy $\mathbf{G}$ o wymiarze $p \times p$. Każda kolumna ϕ_j macierzy $\mathbf{G}$ reprezentuje radialną funkcję bazową z centrum umieszczonym w $\mathbf{x}_j$. Każdy element kolumny ϕ_j jest odpowiedzią funkcji bazowej na pobudzenie x_i zbioru uczącego.

Algorytm zakłada następujące warunki stopu:
- zbiór kandydatów jest pusty,
- błąd sieci jest równy zeru,
- kryterium doboru modelu GVC przestało maleć przez pewną liczbę iteracji algorytmu,
- błąd sieci jest mniejszy od zera z powodu błędów numerycznych.

Szerokość σ funkcji radialnych jest stała dla wszystkich kandydatów. Obliczanie szerokości ze wzoru (3.15) wymaga znajomości liczby neuronów warstwy ukrytej, dlatego rozważany algorytm ustala σ na podstawie wzoru:

$$\sigma = sd_{\max}, \tag{3.22}$$

gdzie s jest współczynnikiem skalującym i może być rozpatrywany jako mianownik wzoru (3.15).

Wybór neuronu, który ma powiększyć strukturę sieci, dokonywany jest na podstawie różnicy pomiędzy błędami sieci o aktualnej strukturze E_m oraz sieci z dodanym neuronem E_{m+1}. Jest on definiowany następująco:

$$E_m - E_{m+1} = \mathbf{y}^T(\mathbf{P}_m^2 - \mathbf{P}_{m+1}^2)\mathbf{y}, \tag{3.23}$$

$$\mathbf{P}_{m+1} = \mathbf{I}_p - \mathbf{G_{m+1}}(\mathbf{G_{m+1}^T G_{m+1}} + \lambda \mathbf{I_m})^{-1}\mathbf{G_{m+1}}^T, \tag{3.24}$$

gdzie $\mathbf{G}_{m+1} = [\mathbf{G}_m \phi_{m+1}]$, ϕ_{m+1} reprezentuje nowy neuron dodany do aktualnego zbioru. Jest to kolumna ze zbioru kandydatów $\mathbf{G}$:

$$\phi_{\mathbf{m+1}} = \begin{bmatrix} \phi_{m+1}(\mathbf{x}_1) \\ \phi_{m+1}(\mathbf{x}_2) \\ \vdots \\ \phi_{m+1}(\mathbf{x}_p) \end{bmatrix}. \tag{3.25}$$

Po podstawieniu do wzoru (3.24) otrzymujemy:

$$\begin{aligned} \mathbf{P}_{m+1} &= \mathbf{I}_p - \begin{bmatrix} \mathbf{G}_m & \phi_{m+1} \end{bmatrix} \\ &\left(\begin{bmatrix} \mathbf{G}_m^T \\ \phi_{m+1}^T \end{bmatrix} \begin{bmatrix} \mathbf{G}_m & \phi_{m+1} \end{bmatrix} + \begin{bmatrix} \lambda_m \mathbf{I}_m & 0 \\ \mathbf{0} & \lambda_{m+1} \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{G}_m^T \\ \phi_{m+1}^T \end{bmatrix} \\ &= \mathbf{I}_p - \begin{bmatrix} \mathbf{G}_m & \phi_{m+1} \end{bmatrix} \\ &\left(\begin{bmatrix} \mathbf{G}_m^T \mathbf{G}_m + \lambda_m & \mathbf{G}_m^T \phi_{\mathbf{m+1}} \\ \phi_{\mathbf{m+1}}^T \mathbf{G}_m & \phi_{m+1}^T \phi_{m+1} + \lambda_{m+1} \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{G_m^T} \\ \phi_{m+1}^T \end{bmatrix} \end{aligned} \tag{3.26}$$

Należy zwrócić uwagę na aspekt obliczeniowy opisywanego algorytmu. Najbardziej wymagającą operacją podczas obliczania błędu jest wyznaczanie macierzy rzutowania i jej odwrotności. Każdy krok algorytmu wymaga zaktualizowania poprzednich wartości macierzy. Ponieważ każdorazowe obliczanie odwrotności całej macierzy jest kosztowne obliczeniowo, stosuje się metody rekurencyjne, które bazują na poprzednich wartościach. Korzystając z własności macierzy, odwrotność będzie miała postać

$$\left(\begin{bmatrix} \mathbf{G}_m^T \mathbf{G}_m + \lambda_m & \mathbf{G}_m^T \phi_{m+1} \\ \phi_{m+1}^T \mathbf{G}_m & \phi_{m+1}^T \phi_{m+1} + \lambda_{m+1} \end{bmatrix} \right)^{-1} = \begin{bmatrix} (\mathbf{G}_m^T \mathbf{G}_m + \lambda_m)^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}$$

$$+\frac{1}{\lambda_{m+1}+\phi_{m+1}^T\mathbf{P}_m\phi_{m+1}}\begin{bmatrix}(\mathbf{G}_m^T\mathbf{G}_m+\lambda_m)^{-1}\mathbf{G}_m^T\phi_{m+1}\\ -1\end{bmatrix}\begin{bmatrix}(\mathbf{G}_m^T\mathbf{G}_m+\lambda_m)^{-1}\mathbf{G}_m^T\phi_{m+1}\\ -1\end{bmatrix}^T \quad (3.27)$$

Ostateczna forma macierzy rzutowania $\mathbf{P}_{m+1}$ powiększona o neuron ma postać:

$$\begin{aligned}\mathbf{P}_{m+1} &= \mathbf{I}_p - \begin{bmatrix}\mathbf{G}_m & \phi_{m+1}\end{bmatrix}\left(\begin{bmatrix}(\mathbf{G}_m^T\mathbf{G}_m+\lambda_m)^{-1} & \mathbf{0}\\ \mathbf{0}^T & 0\end{bmatrix}\right.\\ &+\frac{1}{\lambda_{m+1}+\phi_{m+1}^T\mathbf{P}_m\phi_{m+1}}\begin{bmatrix}(\mathbf{G}_m^T\mathbf{G}_m+\lambda_m)^{-1}\mathbf{G}_m^T\phi_{m+1}\\ -1\end{bmatrix}\\ &\left.\begin{bmatrix}(\mathbf{G}_m^T\mathbf{G}_m+\lambda_m)^{-1}\mathbf{G}_m^T\phi_{m+1}\\ -1\end{bmatrix}^T\right)^{-1}\mathbf{G}_{m+1}^T\\ &= \mathbf{P}_m - \frac{\mathbf{P}_m\phi_{m+1}\phi_{m+1}^T\mathbf{P}_m}{\lambda_{m+1}\phi_{m+1}^T\mathbf{P}_m\phi_{m+1}}. \end{aligned} \quad (3.28)$$

Błąd pomiędzy kolejnymi krokami iteracji algorytmu wyrażony jest wzorem:

$$\begin{aligned}E_m - E_{m+1} &= \mathbf{y}^T(\mathbf{P}_m^2 - (\mathbf{P}_m - \frac{\mathbf{P}_m\phi_{\mathbf{m+1}}\phi_{m+1}^T\mathbf{P}_m}{\lambda_{m+1}\phi_{m+1}^T\mathbf{P}_m\phi_{m+1}})^2)\mathbf{y}\\ &= \frac{2\mathbf{y}^T\mathbf{P}_m^2\phi_{m+1}\mathbf{y}^T\mathbf{P}_m\phi_{m+1}}{\lambda+\phi_{m+1}^T\mathbf{P}_m\phi_{m+1}} - \frac{(\mathbf{y}^T\mathbf{P}_m\phi_{m+1})^2\phi_{m+1}^T\mathbf{P}_m^2\phi_{m+1}}{(\lambda+\phi_{m+1}^T\mathbf{P}_m\phi_{m+1})^2}. \end{aligned} \quad (3.29)$$

W celu przyspieszenia obliczeń wartości zmian w błędzie sieci stosuje się zortogonalizowaną metodę najmniejszych kwadratów [15]. Ułatwia ona obliczanie wkładu poszczególnych kolumn do wartości wyjściowej. Wykorzystanie ortogonalizacji Gram–Schmidta gwarantuje, iż każda nowo dodana kolumna będzie ortogonalna do pozostałych.

Macierz można rozłożyć na iloczyn macierzy z ortogonalnymi kolumnami i macierz trójkątną górną. Macierz $\mathbf{G} \in R^{p\times m}$ będzie miała postać

$$\mathbf{G}_m = \hat{\mathbf{G}}_\mathbf{m}\mathbf{U}_m, \quad (3.30)$$

gdzie $\hat{\mathbf{G}} = [\hat{\phi}_\mathbf{1}\hat{\phi}_\mathbf{2}\cdots\hat{\phi}_\mathbf{m}]$ i $\mathbf{U}$ jest macierzą trójkątną górną.

Po podstawieniu do wzoru na macierz rzutowania $\mathbf{P}_m$ otrzymujemy

$$\mathbf{P}_m = \mathbf{I}_p - \hat{\mathbf{G}}_\mathbf{m}\mathbf{U}_\mathbf{m}(\mathbf{U}_m^T\hat{\mathbf{G}}_\mathbf{m}^T\hat{\mathbf{G}}_\mathbf{m}\mathbf{U}_m + \lambda\mathbf{I}_\mathbf{m})^{-1}\mathbf{U}_m^T\hat{\mathbf{G}}_\mathbf{m}^T. \quad (3.31)$$

Korzystając z własności iloczynu macierzy $(\mathbf{U}_m\mathbf{G}_m^T\mathbf{G}_m\mathbf{U})^{-1} = \mathbf{U}_m^T(\mathbf{G}_m^T\mathbf{G}_m)\mathbf{U}_m$ oraz ortogonalności macierzy $(\mathbf{U}_m)^{-1} = \mathbf{U}_m^T$, $(\mathbf{U}_m^T)^{-1} = \mathbf{U}_m$ oraz $\mathbf{U}_m^T\mathbf{U}_m = \mathbf{I}$ otrzymujemy:

$$\begin{aligned}
\mathbf{P}_m &= \mathbf{I}_p - \hat{\mathbf{G}}_m(\hat{\mathbf{G}}_m^T\hat{\mathbf{G}}_m + \lambda\mathbf{I}_m)^{-1}\hat{\mathbf{G}}_m^T \\
&= \mathbf{I}_p - \hat{\mathbf{G}}_\mathbf{m}\begin{bmatrix} \frac{1}{\lambda+\hat{\phi_1}^T\hat{\phi_1}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda+\hat{\phi_2}^T\hat{\phi_2}} & 0 & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\lambda+\hat{\phi_m}^T\hat{\phi_m}} \end{bmatrix}\hat{\mathbf{G}_m^T} \qquad (3.32) \\
&= \mathbf{I}_p - \sum_{j=1}^{m}\frac{\hat{\phi_j}\hat{\phi_j^T}}{\lambda + \hat{\phi_j^T}\hat{\phi_j}}.
\end{aligned}$$

Dodanie kolejnego neuronu powoduje dodanie kolejnego elementu do sumy

$$\mathbf{P}_{m+1} = \mathbf{I}_p - \sum_{j=1}^{m}\frac{\hat{\phi_j}\hat{\phi_j^T}}{\lambda + \hat{\mathbf{h}}_j^T\hat{\mathbf{h}}_j} - \frac{\hat{\mathbf{h}}_{m+1}\hat{\mathbf{h}}_{m+1}^T}{\lambda + \hat{\mathbf{h}}_{m+1}^T\hat{\mathbf{h}}_{m+1}} \qquad (3.33)$$

Zmiana w wartości błędu (3.29) przed i po dodaniu neuronu wyraża się wzorem:

$$E_m - E_{m+1} = \frac{(\mathbf{y}^T\hat{\mathbf{h}}_{m+1})^2 2\lambda + \hat{\mathbf{h}}_{m+1}^T\hat{\mathbf{h}}_{m+1}}{(\lambda + \hat{\mathbf{h}}_{m+1}^T\hat{\mathbf{h}}_{m+1})^2}. \qquad (3.34)$$

Zortogonalizowany wektor wag zapisywany jest następująco:

$$\begin{aligned}
\hat{\mathbf{w}}_m &= (\hat{\mathbf{G}}_m^T\hat{\mathbf{G}}_m + \lambda\mathbf{I}_m)^{-1}\hat{\mathbf{G}}_m^T\mathbf{y} \\
&= \begin{bmatrix} \frac{1}{\lambda+\hat{\phi_1}^T\hat{\phi_1}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda+\hat{\phi_2}^T\hat{\phi_2}} & 0 & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\lambda+\hat{\phi_m}^T\hat{\phi_m}} \end{bmatrix}\begin{bmatrix} \hat{\mathbf{h}}_1^T \\ \hat{\mathbf{h}}_2^T \\ \vdots \\ \hat{\mathbf{h}}_\mathbf{m}^\mathbf{T} \end{bmatrix}\mathbf{y} \qquad (3.35) \\
&= \sum_{j=1}^{m}\frac{\mathbf{y}^T\hat{\mathbf{h}}_j}{\lambda + \hat{\mathbf{h}}_j^T\hat{\mathbf{h}}_j}.
\end{aligned}$$

Korzystając ze wzoru (3.9), wektor wag opisany równaniem (3.35) ostatecznie przyjmuje postać:

$$\begin{aligned} \mathbf{w}_m &= (\mathbf{G}_m^T\mathbf{G}_m + \lambda\mathbf{I}_m)^{-1}\mathbf{G}_m^T\mathbf{y} \\ &= \mathbf{U}_m^{-1}(\hat{\mathbf{G}}_m^T\hat{\mathbf{G}}_m + \lambda\mathbf{I}_m)^{-1}\hat{\mathbf{G}}_m^T\mathbf{y} \\ &= \mathbf{U}_m^{-1}\hat{\mathbf{w}}_m. \end{aligned} \tag{3.36}$$

3.2.9. Przykład obliczeniowy - modelowanie układu Lorenza

Poprawność zaproponowanego algorytmu 1 modelowania sprawdzono na układzie Lorenza, opisanego równaniami różniczkowymi w postaci:

$$\begin{aligned} \frac{dx(t)}{dt} &= -\sigma_L x(t) + \sigma_L y(t), \\ \frac{dy(t)}{dt} &= rx(t) - x(t)z(t) - y(t), \\ \frac{dz(t)}{dt} &= x(t)y(t) - bz(t), \end{aligned} \tag{3.37}$$

Przyjęto następujące wartości parametrów $\sigma_L = 10$, $b = 8/3$, $r = 28$.

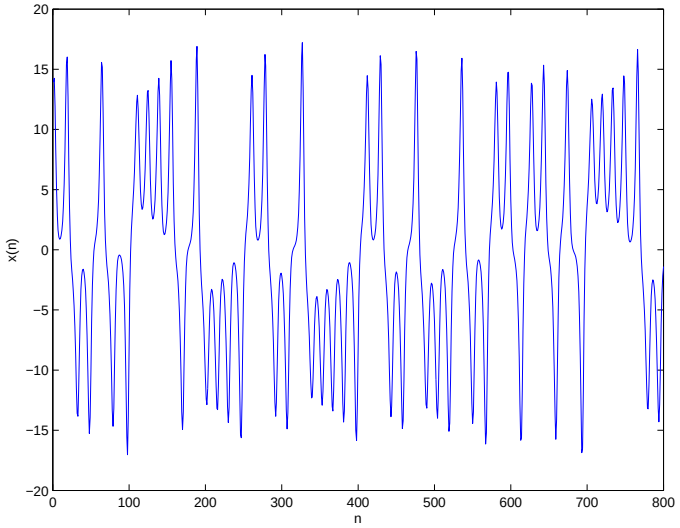
Chaos deterministyczny w układzie Lorenza występuje, gdy parametr r (parametr Rayleigha) przekroczy wartość krytyczną $r \approx 24,74$. Szczegółowe zmiany zachowań układu Lorenza w zależności od wartości parametrów opisane są w książce [51] (rozdz. 12).

Układ o powyższych parametrach charakteryzuje się niecałkowitym wymiarem fraktalnym. Jest on również wrażliwy na warunki początkowe i ma dodatnie wykładniki Lapunowa, co czyni dynamikę układu nieprzewidywalną w dłuższym czasie.

Równanie (3.37) rozwiązano metodą Rungego–Kutty ze stałym krokiem $0,05$. Warunki początkowe dla metody całkowania wynosiły $x_0 = 0,2, y_0 = 0,1, z_0 = 1$. Wybrano wartość kroku całkowania według pracy [43]. Autor twierdzi, że jest ona wystarczająca do poprawnego modelowania układu Lorenza. Eksperymenty opisane dalej potwierdzają tę tezę.

Składowa x układu Lorenza traktowana jest jako szereg czasowy powstały z wyjścia układu. Jej wykres przedstawiono na rysuneku 3.1. Na jego podstawie rekonstruowana jest dynamika i modelowany układ. Dostępne informacje o układzie ograniczone są do szeregu czasowego. Zakładany jest brak dodatkowej wiedzy o własnościach obiektu.

Do zrekonstruowania dynamiki układu na podstawie dostępnego szeregu czasowego użyto teorii rekonstrukcji. Parametrami rekonstrukcji dynamiki są wymiar zanurzenia n i opóźnienie τ. Zgodnie z twierdzeniem Takensa [53], aby zagwarantować poprawną rekonstrukcję przestrzeni stanów, wymiar zanurzenia powinien



Rysunek 3.1. Jednowymiarowe wyjście równań Lorenza

być dwukrotnie większy od wymiaru fraktalnego układu. Wymiar korelacyjny oryginalnego szeregu wynosił $d_2 = 2,16$, czyli minimalny wymiar zanurzenia powinien spełniać warunek $n \geqslant 7$. Wymiar zanurzenia wyznaczono również metodą fałszywych sąsiadów [29].

Uwzględniając rezultat metody fałszywych sąsiadów oraz zakładając, że można odtworzyć dynamikę układu w wymiarze zanurzenia równym wymiarowi topologicznemu, za minimalny wymiar zanurzenia przyjęto $n = 3$. Wymiar był zwiększany w przypadku, gdy zbudowany model niepoprawnie odwzorowywał dynamikę układu.

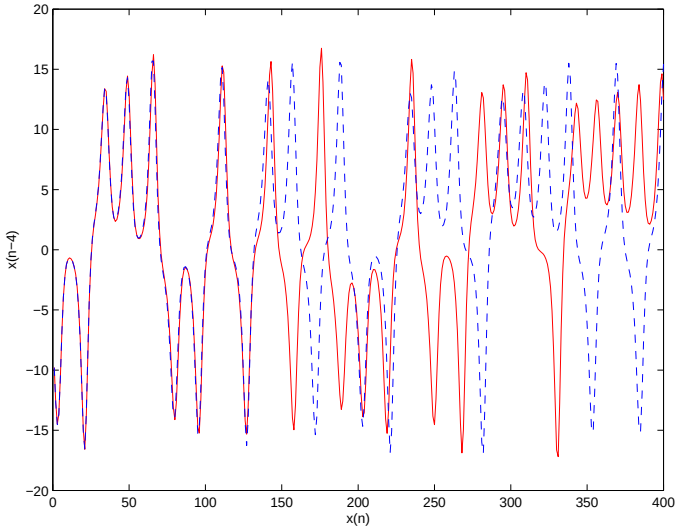
Opóźnienie wyznaczono, korzystając z metody wzajemnych informacji [18] i wynosiło ono $\tau = 4$. Portret opóźnień szeregu czasowego pokazany jest na rysunku 3.3.

Zbiór uczący powstał z oryginalnego szeregu (wzór (3.17) lub (3.18)), czyli opóźnienie przyjmowało wartości ze zbioru $\tau = \{1, 4\}$.

W zależności od ustalonych parametrów n i τ, liczba wejść sieci mieści się w przedziale $(3; 12)$. Eksperymenty wykazały, że najlepszym, ze względu na jakość predykcji mierzonej MSE, jest ciąg uczący zbudowany według wzoru (3.18) o opóźnieniu i zanurzeniu równym odpowiednio $\tau = 4$ i $n = 3$.

Wartość parametru regularyzacji została ustalona na $\lambda = 10^{-6}$ i nie była zmieniana w trakcie eksperymentów.

Do budowy modelu układu użyto zbioru uczącego o liczności $p = 3000$ elemen-



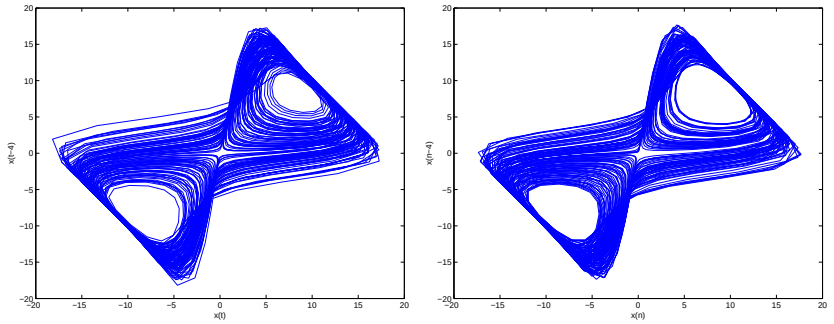
Rysunek 3.2. Wykres oryginalnego szeregu czasowego (linia ciągła) i predykowanego przez sieć neuronową (linia przerywana)

tów. Dokładność predykcji i odtworzenia dynamiki przez strukturę sieci szacowano na zbiorze rekurencyjnie predykowanym przez sieć o długości równej próbce uczącej. Wykres generowanego przez sieć szeregu czasowego oraz odpowiadający mu wykres szeregu czasowego układu znajdują się na rysunku 3.2. Model stosunkowo dokładnie odwzorowuje układ przez pierwszych kilkudziesięciu krokach, po czym wartości się rozbiegają. Spowodowane jest to chaotycznym charakterem układu. Czas, po którym predykcja się załamuje, jest zgodny z horyzontem predykcji opisanym wzorem 3.1. Portret opóźnień dla wygenerowanego przez sieć szeregu czasowego oraz odpowiadającego mu szeregu układu znajdują się na rysunku 3.3.

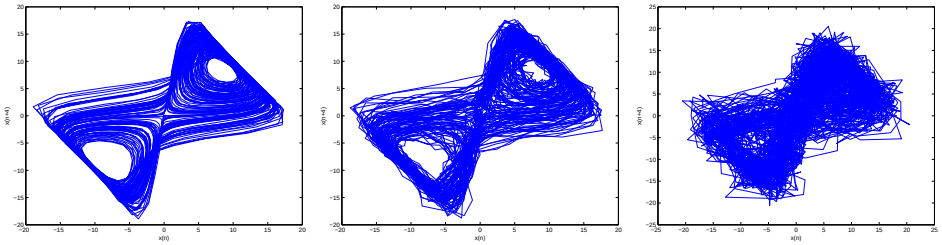
Uzyskane wyniki predykcji są podobne lub lepsze od wyników uzyskanych w pracach [24], [45]. Należy również podkreślić, że nie stosowano metod reestymacji wartości regularyzacji jak na przykład w cytowanych pracach.

3.2.10. Modelowanie z szumem

W dalszych eksperymentach sprawdzono możliwość odtworzenia dynamiki w przypadku dodania gaussowskiego szumu do próbki uczącej. Stosunek SNR



Rysunek 3.3. Od lewej: portret opóźnień oryginalnego szeregu, portret opóźnień szeregu wygenerowanego przez sieć



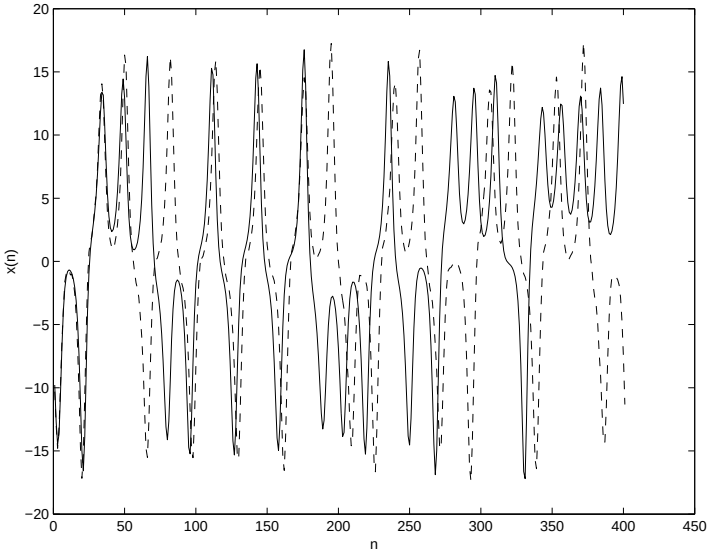
Rysunek 3.4. Zaszumiony atraktor Lorenza odpowiednio $SNR = 30, 20, 15$

(ang. *signal to noise ratio*) sygnału do szumu wyrażony jest wzorem:

$$SNR = 10log_{10}\Big[\frac{\sigma^2_{sygnal}}{\sigma^2_{szum}}\Big] \tag{3.38}$$

Przeprowadzono eksperymenty dla wartości $SNR = 30, 20, 15$. Z powodu zbyt wysokiego poziomu szumu nie udało się zamodelować układu dla ciągu uczącego o $SNR = 10$. Na rysunku 3.4 znajdują się portrety opóźnień dla poszczególnych wartości SNR. Metoda uczenia, wartość mnożnika, wartość parametru regularyzacji λ oraz długość ciągu uczącego zostały ustalone jak dla przypadku bez szumu. Nauczona sieć inicjowana ostatnim wejściem ciągu uczącego predykowała rekurencyjnie 3000 kolejnych wartości.

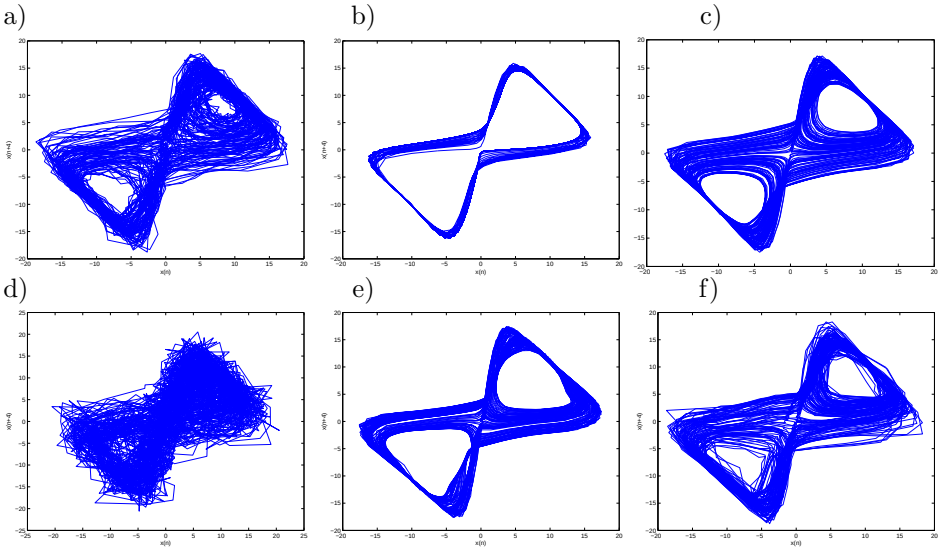
Wykres predykowanego szeregu czasowego powstał dla $SNR = 15$, a odpowiadający mu szereg czasowy układu znajduje się na rysunku 3.5. Predykowana trajektoria rozbiega się o wiele szybciej niż w przypadku, gdy model tworzony był z użyciem ciągu bez szumu (rys. 3.2).



Rysunek 3.5. Oryginalny szereg i predykowany przez model powstały na zbiorze uczącym o $SNR = 15$

Zaszumienie sygnału dla $SNR = 30$ jest znikome i nie powoduje problemów z rekonstrukcją dynamiki. Jednak przy większym zaszumieniu modelowanie z parametrami przyjętymi dla przypadku bez szumu jest niemożliwe. Zaobserwowano, że poprawność modelu silnie zależy od wymiaru zanurzenia. Algorytm konstrukcyjny przy zwiększającym się zaszumieniu i przy $n = 3$ wygenerował zachowanie okresowe, rysunek 3.6b. Zwiększenie wymiaru do $n = 4$ poprawiło jakość modelu – rysunki 3.6c i 3.6f. Należy jednak zwrócić uwagę, że zbyt duża wartość wymiaru zanurzenia prowadzi do nadmiernego dopasowania do danych, co częściowo obserwujemy dla $n = 6$ i $SNR = 15$ (rys. 3.6f).

Ponieważ poprawna predykcja dla układów chaotycznych na dłuższym przedziale czasu jest niemożliwa, jakość modelu sprawdzono również za pomocą niezmienników dynamiki. Niezmiennik dynamiki jest wartością charakterystyczną, która nie zmienia się w trakcie działania (ewolucji) układu. Takim niezmiennikami są między innymi wymiar fraktalny oraz wykładniki Lapunowa. W tabeli 3.1 zestawione są wartości wykładników Lapunowa λ_L oraz estymacji wymiaru fraktalnego Kaplana–Yorka dla różnych wartości zaszumienia i wymiaru zanurzenia. Wartości wymiaru zgadzają się z wymiarem atraktora Lorenza obliczonym w [21]. Wartości wykładników Lapunowa dla wszystkich wartości SNR są dodatnie, co wskazuje na chaotyczną naturę wygenerowanego przez model szeregu.

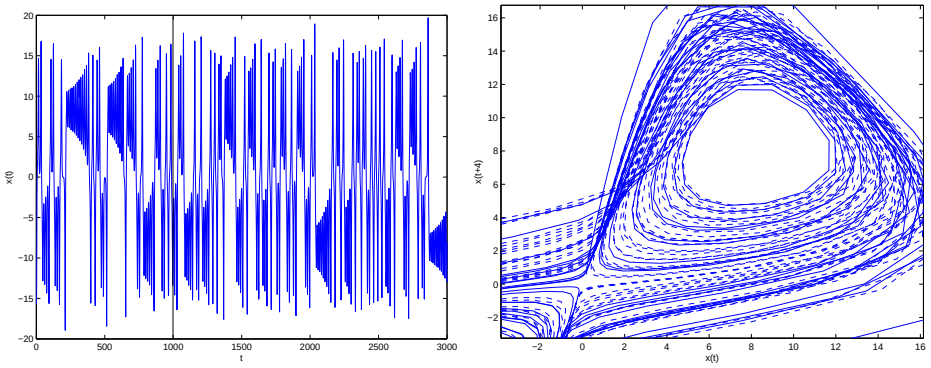


Rysunek 3.6. Pierwszy wiersz odpowiednio: od lewej atraktor Lorenza $SNR = 20$, zrekonstruowany atraktor dla $n = 3$ i $n = 4$. Drugi wiersz odpowiednio: atraktor Lorenza o $SNR = 15$, zrekonstuowany atraktor dla $n = 4$ i $n = 6$

Tabela 3.1. Wartości największego wykładnika Lapyunova oraz wymiaru Kaplana–Yorka dla różnych wartości SNR i wymiarów zanurzenia.

SNR	λ_L	$D_{KY} n = 4$	$D_{KY} n = 5$	$D_{KY} n = 6$
–	1,5	2,050		
30	1,4	2,033	2,028	2,031
20	1,4	1,887	2,001	1,918
15	1,5	2,008	1,619	1,396

(a) (b)



Rysunek 3.7. (a) Wykres fragmentu szeregu czasowego zawierającego małą zmianę w układzie Lorenza w chwili $t = 1000$ przy zmianie parametru r z 28 na 29.4. Pionowa linia oznacza moment wystąpienia zmiany.
(b) Portret opóźnień. Linia ciągła – trajektoria układ przed zmianą, linia przerywana – trajektoria po wystąpieniu zmiany

3.3. Wykrywanie zmian w dynamice układu chaotycznego

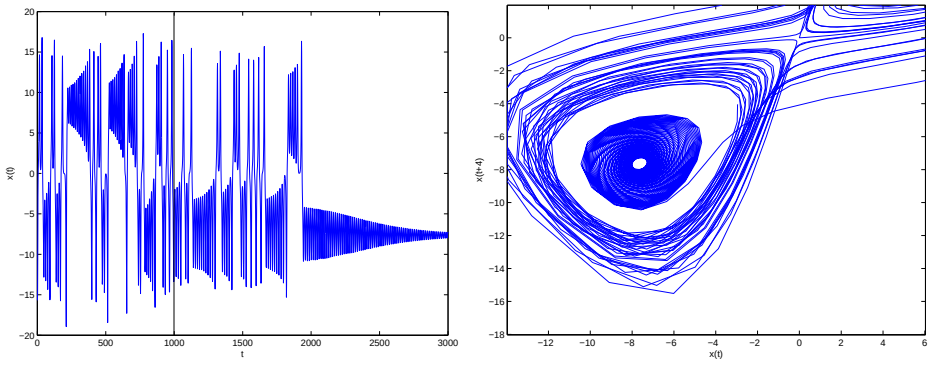
W układach chaotycznych możemy wyróżnić dwie grupy zmian typu:
- małej zmiany – nie powoduje zmiany typu zachowania układu, np. układ jest chaotyczny i po wystąpieniu takiej zmiany pozostaje chaotyczny,
- drastycznej zmiany – powoduje zmianę typu zachowania układu, np. układ jest chaotyczny po wystąpieniu zmiany staje się okresowym.

Wykres szeregu czasowego reprezentującego drastyczną zmianę przedstawiony jest na rysunku 3.8a. Zmiana jest dobrze widoczna i powoduje przejście do innego typu dynamiki układu. Rezultatem jest przejście z atraktora chaotycznego do okresowego. Portret opóźnień obrazujący zbieganie atraktora znajduje się na rysunku 3.8b.

Wykres szeregu czasowego zawierającego małą zmianę pokazany jest na rysunku 3.7a. Zmiana dynamiki spowodowana tą zmianą nie powoduje widocznych zmian w atraktorze. Zastosowanie portretu opóźnień (rys. 3.7b również nie poprawia rozpoznawalności zmiany. Trajektoria po zmianie wygląda jak dalsza ewolucja układu z niezmienioną dynamiką.

Poprzedni podrozdział 3.2 pokazał, że zaproponowany model układu chaotycznego, oparty na sieci RBF, dokładnie odtwarza dynamikę układu chaotycznego. Korzystając z tego modelu możliwe jest opracowanie algorytmu wykrywania zmian.

(a) (b)



Rysunek 3.8. (a) Wykres fragmentu szeregu czasowego zawierającego drastyczną zmianę w układzie Lorenza w chwili $t = 1000$ przy zmianie parametru r z 28 na 22.9. Pionowa linia oznacza moment wystąpienia zmiany.
(b) Portret opóźnień. Linia ciągła – trajektoria układ przed zmianą, linia przerywana – trajektoria po wystąpieniu zmiany

Schemat blokowy wykrywania zmian oparty na modelu pokazany jest na rysunku 3.9. Blok modelu realizowany jest przez sieć neuronową RBF. Cechami są residua obliczone jako różnica pomiędzy bieżącą wartością wyjścia układu a wartością predykowaną przez model. Natomiast w bloku wykrywania zmian użyto funkcji progowej (3.39).

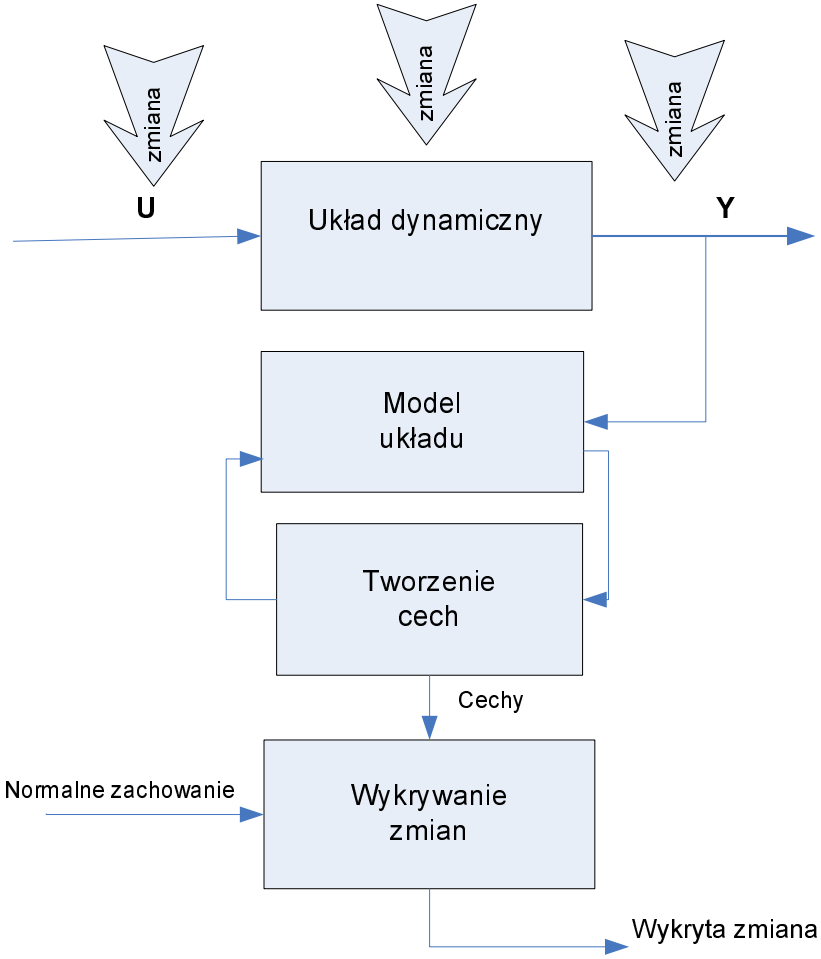
Prostym kryterium wykrywania zmian korzystającym z wartości charakterystycznych jest użycie progu. Jeżeli wartość cechy przekroczy zadany próg h, wykrywana jest zmiana.

$$\text{decyzja w momencie } t = \begin{cases} \text{zmiana} & \textit{cecha} \geqslant h, \\ \text{brak zmiany} & \textit{cecha} < h. \end{cases} \tag{3.39}$$

Zastosowanie residuów umożliwia wyróżnienie dwóch podejść do wykrywania zmian:

– zastosowanie funkcji progowej, dla której argumentem są residua,
– zastosowanie funkcji progowej, dla której argumentem jest suma residuów w oknie o długości k.

Algorytm 4. *Wykrywania zmian oparty na modelu:*
1. Zbuduj model zgodnie z algorytmem 3.



Rysunek 3.9. Schemat blokowy wykrywania zmian z użyciem modelu

2. *Wyznacz residuum $e(t)$ w chwili t pomiędzy wyjściem układu a wartością predykowaną przez model:*

$$e(t) = y_t - \sum_{i=1}^{m} w_i \phi_i(\mathbf{x_t}), \tag{3.40}$$

gdzie y_t jest wyjściem modelu, natomiast suma po m neuronach wewnętrznych reprezentuje wejście modelu.
3. *Monitoruj residua $e(t)$ lub sumę residuów obliczonych w ruchomym oknie o długości k, $\sum_{i=0}^{k-1} e(t+i)$.*
4. *Jeżeli wartości residuów lub ich sumy przekroczą zadany próg h, zasygnalizuj zmianę. W przeciwnym wypadku wróć do kroku 3.*

Kluczowym elementem w tym algorytmie jest dokładność predykcji modelu. Należy przypomnieć, iż wrodzoną własnością układów chaotycznych jest wrażliwość na warunki początkowe, objawiająca się szybkim rozbieganiem dwóch blisko położonych trajektorii w przestrzeni stanów. W kontekście predykcji kolejnych wartości wyjścia układu skutkuje to krótkim czasem, po którym dokładność się załamuje. Horyzont czasowy zależy od wartości wykładnika Lapunowa i opisany jest zależnością (3.1). W celu zachowania poprawności predykcji przez dłuższy czas niezbędne jest cykliczne inicjowanie (restartowanie) modelu bieżącym wyjściem układu. Dobór wartości, po której następuje restart, jest uzależniony od horyzontu predykcji lub eksperymentalnie.

W celu uproszczenia dalszych rozważań, liczba predykowanych przez model wartości, po których następuje restart jest równa k długości, okna na którym sumowane są residua.

Decyzja o wystąpieniu zmiany podejmowana jest na podstawie wartości funkcji progowej (3.39), której argumentem są residua. Wartość progu h dobierana jest eksperymentalnie. Zamiast prostej funkcji progowej zalecane jest stosowanie kart kontrolnych do detekcji zmiany.

W rozważanym przypadku na kartę kontrolną nanoszone są wartości residuów lub ich sumy. Zalecane jest użycie karty działającej na pojedynczych wartościach, np. EWMA [46] dla małych zmian lub karty Schewart [30] dla zmian drastycznych.

3.3.1. Przykład obliczeniowy – wykrywanie zmian w układzie Lorenza

Przykład poprawności działania algorytmu pokazano dla układu Lorenza opisanego równaniem (3.37). Model został zbudowany według metody opisanej w rozdziale 1. Ciąg uczący powstał, zgodnie z podrozdziałem 3.2.6, z szeregu czasowego reprezentującego wyjście układu Lorenza wolne od zmian. Nauczony model zainicjowano, a następnie rekurencyjnie predykowano kolejne wartości. Ze względu

na krótki czas dokładnej predykcji, model był restartowany co k kroków przez aktualną wartość wyjścia układu.

Zmiana została wprowadzona do układu za pomocą parametru r równania Lorenza (3.37). Zakres wartości, w którym dokonywano zmian, nie wpływał na chaotyczny charakter układu.

W celu zapobiegnięcia błędnej analizie, wynikającej z dobrego dopasowania modelu do danych uczących, szereg użyty do budowy modelu powinien być różny od szeregu zawierającego zmianę.

Za przykłady działania proponowanego algorytmu posłużą dwa eksperymenty, w których wprowadzono dwie następujące po sobie małe zmiany o różnych wartościach. Na rysunku 3.10 (prawa kolumna) pokazano szereg czasowy z 20% zmianami parametru $r : 28 \rightarrow 33,6$ i $r : 33,6 \rightarrow 28$. Pierwsza zmiana następuje w chwili czasowej $t = 100$, kolejna w chwili $t = 200$. Zmiany o takiej wielkości rozważane są w pracy [25].

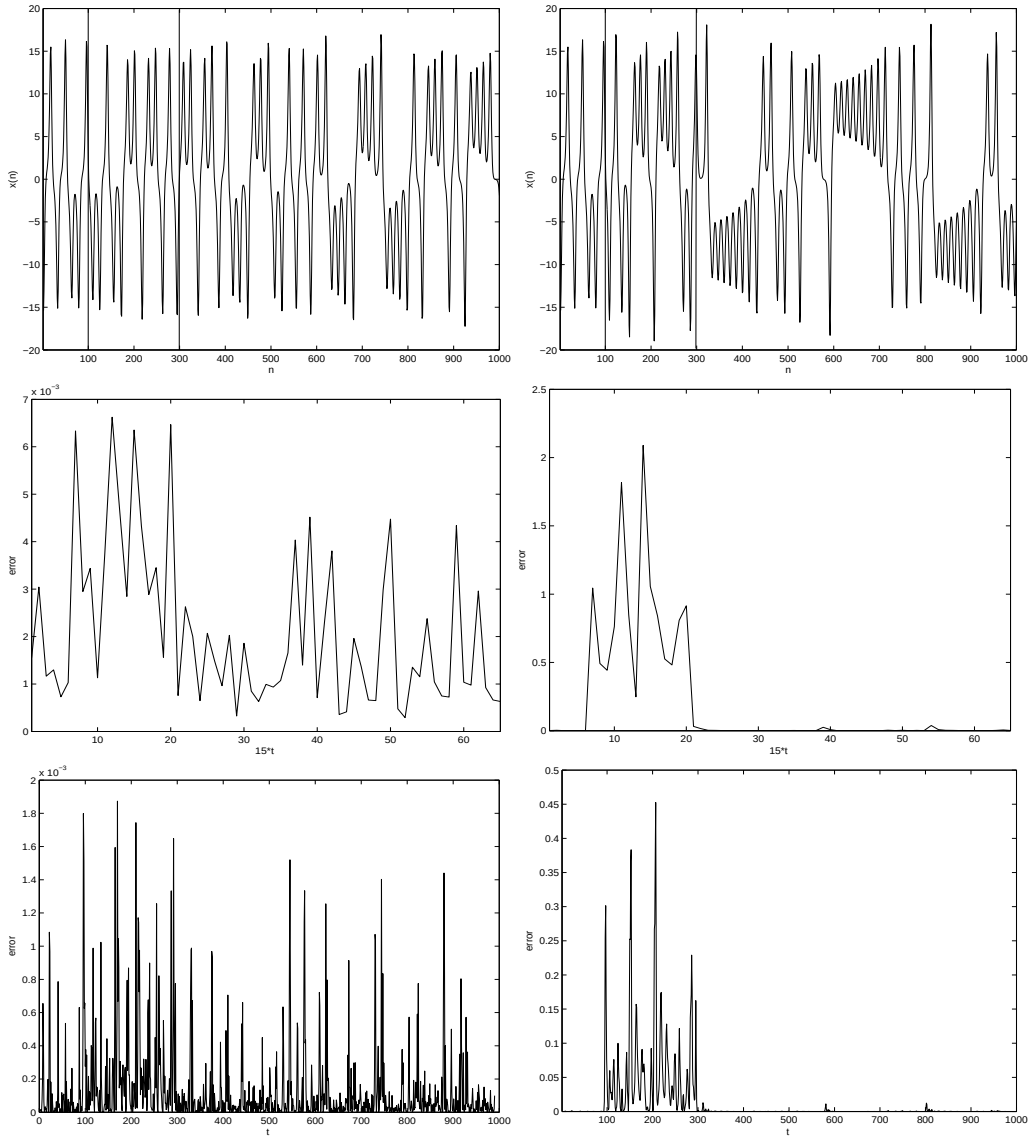
W drugim eksperymencie wprowadzono zmiany mniejsze o rząd wielkości od zmian rozważanych w pierwszym przykładzie. Lewa kolumna, rysunku 3.10 pokazuje następujące po sobie dwie 2% zmiany parametru $r : 28 \rightarrow 28,56$ i $r : 28,56 \rightarrow 28$. Zmiany nastąpiły w chwili $t = 100$ i $t = 200$. Należy podkreślić, że tak małe zmiany nie były dotychczas rozważane w literaturze.

Przeprowadzono predykcję w oknie o długości $k = 15$. Po tym czasie model był ponownie inicjowany wyjściem układu. Wykres wartości residuów dla poszczególnych chwil czasowych analizowanych szeregów pokazano w ostatnim wierszu rysunku 3.10. Natomiast wykres wartości sumy residuów znajduje się w wierszu środkowym rysunku.

Dla pierwszego przykładu analiza wykresu, zarówno residuów, jak i ich sumy, nie pozostawia wątpliwości co do miejsca wystąpienia zmian. Jednak w przypadku 2% wystąpienie zmiany nie jest już takie oczywiste. Wartości residuów w obszarze zmiany są zbliżone do wartości residuów dla działającego poprawnie układu. Zastosowanie zbyt małego progu h prowadzi do powstania fałszywych alarmów. Rozwiązaniem tego problemu jest użycie sumy residuów, powodując uzyskanie wartości ułatwiających identyfikację zmiany.

Z przeprowadzonych eksperymentów wynika, iż wprowadzane małe zmiany, powyżej 6% oryginalnej wartości parametru, są łatwo wykrywalne z użyciem residuów, jak i ich sumy. Zmiany poniżej 6% wykrywalne są jedynie z użyciem sumy residuów obliczonych w oknie o długości k. Zmian mniejszych niż 2% nie udało się poprawnie wykryć metodą z modelem RBF.

Długość okna k ma znaczący wpływ na szybkość wykrycia zmiany i liczbę fałszywych alarmów. Im dłuższe okno, tym później zmiana zostanie wykryta. Eksperymentalnie stwierdzono, że zadowalające wyniki otrzymuje się dla $k \in (10; 26)$.



Rysunek 3.10. Lewa kolumna: szereg czasowy z 2% zmianami dla chwili $t = 100$ i $t = 200$; wartości residuów sumowanych w oknie $k = 15$; wartości residuów. Prawa kolumna: szereg czasowy z 20% zmianami dla chwili $t = 100$ i $t = 200$; wartości residuów sumowanych w oknie $k = 15$; wartości residuów

Właściwy dobór wartości parametrów procesu jest bardzo istotny, jeśli idzie o szybkość i dokładność wykrywania zmian. Długość okna ma wpływ zarówno na czułość, jak i czas wykrycia zmiany. W praktyce dobór parametrów zależy od przetwarzanego szeregu.

3.4. Podsumowanie

W rozdziale opisano metodę wykrywania zmian w układach chaotycznych na podstawie skalarnego strumienia danych utworzonego z szeregu czasowego obserwacji wyjścia. Wykryta zmiana traktowana jest jako bodziec lub ostrzeżenie. Rozpatrywana jest również jako element prediagnostyczny, który (bez matematycznego opisu zjawiska) odpowiada na pytanie czy w układzie nastąpiła, czy też nie nastąpiła zmiana.

Pierwsza cześć rozdziału zawiera zaproponowaną metodę modelowania dynamiki układów chaotycznych. Zapisano ją w formie algorytmu 1.

Model został zbudowany z wykorzystaniem sieci neuronowej RBF i opisany algorytmem 3. Zbiór uczący, jak również liczba wejść sieci zostały wyznaczone przez procedury rekonstruujące przestrzeń stanu. Sprawdzono poprawność predykcyjną modelu zbudowanego za pomocą algorytmu konstrukcyjnego selekcji postępującej dla zaszumionych i niezaszumionych zbiorów uczących. Jakość modelu została zweryfikowana również z użyciem niezmienników dynamiki, tj. wymiaru fraktalnego i maksymalnego wykładnika Lapunowa. Nawet przy wysokim poziomie szumu w zbiorze uczącym model poprawnie odtwarza dynamikę układu.

Algorytmy konstrukcyjne nie były dotychczas prezentowane w literaturze jako metody tworzenia modeli układów chaotycznych. Ich podstawową zaletą, oprócz dokładnej predykcji, jest ustalenie struktury modelu podczas procesu uczenia oraz brak konieczności reestymacji parametru regularyzacji.

W drugiej części rozdziału przedstawiono metodę wykrywania zmian w strumieniach danych pochodzących z układów chaotycznych bazującą na modelu. Wykrywa ona zarówno małe, jak i drastyczne zmiany w dynamice układu.

Układy chaotyczne charakteryzują się krótkim horyzontem predykcji, po którego przekroczeniu dokładność drastycznie się załamuje. Jednak użycie w metodzie cyklicznej reinicjalizacji modelu wyjściem układu oraz mały błąd predykcji zapewnia poprawne modelowanie układu przez dostatecznie długi czas, który umożliwia realizację zadania wykrywania zmian. W celu wykrywania małych zmian zaproponowano sumowanie residuów na ruchomym oknie o stałej długości.

Metodę przetestowano, modyfikując wartości parametrów układu Lorenza. Wykrywa ona małe zmiany przekraczające 2% wartości oryginalnego parametru. Metoda poprawnie wykrywa również sekwencje następujących po sobie zmian.

Działanie algorytmu modelowania i wykrywania zmian pokazano na przykładach obliczeniowych.

Bibliografia

[1] Abarbanel H.D., *Tools for the analysis of chaotic data*, Fildes Institute Communication, 36(11), 1997.

[2] Abarbanel H.D., Frison H.D., Tsimiring H.D. *Obtaining order in a world of chaos: Time-domain analysis of nonlinear and chaotic signals*, IEEE Signal Processing, (98), 1998.

[3] Abarbanel H.D., Brown R., Kennel M.B. *Lyapunov exponents in chaotic systems: Their importance and their evaluation using observed data*, International Journal of Modern Physics B, 5(9):1347 – 1375, 1991.

[4] Ashrafi S., Roszman L. *Nonlinear techniques for forecasting solar activity directly form its time series*, Technical report, NASA, 1992.

[5] Benoudjit N., Archambeau C., Lendasse A. *Width optimization of the gaussian kernels inradial basis function networks*, In European Symposium on Artificial Neural NetworksBruges, 2000.

[6] Bishop C.M., *Improving the generalization properties of radial basis function neural networks*, Neural Computation, 3(4), 1991.

[7] Broomhead D.S., Lowe D., *Multivariable functional interpolation and adaptive networks*, Complex Systems, 2:269–303, 1988.

[8] Bryant P., Brown R., *Lyapunov exponents from observed time series*, Physical Review Letters, 65(13), 1990.

[9] Casdagli M., *Nonlinear prediction of chaotics time series*, Physica D, 35, 1989.

[10] Celebi M.E., Koivo H.N., Ukyan Z., Guezelis C., *Analysis of ipnput-output clustering for determining centers of rbfn*, IEEE Trans. Neural Network, 11(4), 2000.

[11] Chen H., Chen T., *Approximation capability to functions of serveral variables nonlineat functionals and operators by radial basis function neural network*, IEEE Transaction on Neural Networks, 6:904–910, 1995.

[12] Cowper M.R., Mulgrew B., Unsworth C.P., *Nonlinear prediction of chatotic singals using normalised radial basis function network*, Singal processing, 82:775–789, 2002.

[13] Girosi F., Poggio T., *Networks for approximation and learning*, IEEE proceedings, 9, 1990.

[14] Giordana A., Blanzieri A., Katenkamp P., *Growing radial basis function networks*, 1995.

[15] Grant P.M., Chen S., Cowam C.F.N., *Orthogonal least squares learning for radial basis function networks*, IEEE Transaction on Neural Networks, 2:302–309, 1991.

[16] Eksioglu B., Demirer R., Capar I., *Subset selection in multiple linear regression: a new mathematical programming approach*, Comput. Ind. Eng., 49(1):155–167, 2005.

[17] Farmer J.D., Sidorowich J., *Predicting chaotic time series*, P. Rev. Lett., 59(8), 1987.

[18] Fraser A.M., Swinney H.L., *Independent coordinates for strange attractors from mutual information*, Phys. Review A, 33:1134–1140, February 1986.
[19] Fritzke B., *Growing cells structures – a self-organizing network for unsupervised and supervised learning*, Neural Computation, 9(7), 1994.
[20] Grassberger P., *An optimized box-assisted algorithm for fractal dimensions*, Physics Letters A, 148:63–68, August 1990.
[21] Grassberger P., Procaccia I., *Measuring the strangeness of strange attractor*, Physica D, 9:189–208, 1983.
[22] Haykin S., *Neural networks : a comprehensive foundation*, Prentice Hall, 1999.
[23] Haykin S., Yee P., *Regularized Radial Basis Function Network*, John Wiley, 2001.
[24] Haykin S., Principe J., *Making sense of complex world*, IEEE Signal Process. Mag., 15:66–81, 1998.
[25] Ilin A., Valpola H., *Nonlinear dynamical factor analysis for state change detection*, IEEE Trans. on Neural Network, 15(3):559–575, 2004.
[26] Jankowski N., Duch W., *Survey of neural transfer functions,* Neural Computing Surveys, 2:163–212, 1999.
[27] Kannathal N., Puthusserypady S.K., Choo M.L., *Elman neural networks for dynamic modeling of epileptic eeg*, In Engineering in Medicine and Biology Society, 2006.
[28] Kaminski W., Strumillo P., *Neural networks with orthogonalised transfer functions*, 2001.
[29] Kennel M.B., Brown R., *Determining embedding dimension for phase-space reconstruction using a geometrical construction*, Phys.Rev. A, 45:3403–3411, 1992.
[30] Kramer H., *On control Charts for Time Series*, PhD thesis, Universitat Ulm, 1997.
[31] Lai D., Chen G., *Statistical analysis of lyapunov exponents from time series: A jacobian approach*, Mathl. Comput. Modelling, 27(7).
[32] Mess A., Judd K., *On selecting models for nonlinear time series*, Physica D, 82:426–444, 1995.
[33] Manzan S., *Model selection for nonlinear time series*, Empirical Economics, 29:901–920, 2004.
[34] Miguel F., Acosta A., *Radial basis function and related models: An overwiew*, Signal processing, 45:37–58, 1995.
[35] Moody J., Darken C.J., *Fast learning in networks of locally-tuned processingunits*, Neural Computatiom, 1:281–294, 1989.
[36] Moody J.E., *The effective number of parameters: An analysis of generalisation and regularisation in nonlinear learning systems.* Morgan Kaufmann, San Mateo, 1992.
[37] Niranjan M., Kadirkamanathan V., *A function estimation approach to sequential learning with neural networks*, Neural Computation, 5(6), 1993.
[38] Orr M.J., *Regularisation in the selection of rbf centres*, Neural Computation, 7(3), 1995.
[39] Orr M.J., *Introduction to radial basis function networks*, Technical report, University of Edinburgh, 1996.
[40] Orr M.J., *Recent advances in radial basis function networks*, 1999. Technical report, University of Edinburgh, 1999.

[41] Packard N., Crutchfield J.P., Farmer J.F, Shaw R.S., *Geometry from a time series*, Physical Review Letters, 45:712–716, 1980.

[42] Platt J., *A resource-allocating network for function interpolation*, Neural Computation, 3(2), 1991.

[43] Principe J.C., Wang L., Motter M.A., *Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control*, In Proc. of the IEEE, 1998

[44] Rafajłowicz E., Skubalska-Rafajłowicz E., *Rbf nets based on equidistributed point*, In Proc. of the 9th IEEE 9th IEEE Int. Conference on Methods and Model, 2003.

[45] Rank E., *Application of bayesioan trained rbf networks to nonlinear time-series modeling*, Signal processing, 83:1393–1410, 2003.

[46] Roberts S.W., *Control chart tests based on geometric moving avarages*, Technometrics, (1):239–250, 1959.

[47] Sandberg W., Park J., *Universal approximation using radial-basis functions*, Nerual Computation, 3:246–257, 1991.

[48] Sawada Y., Sano M., *Measurement of the lyapunov spectrum from a chaotic time series*, Phys. Rev. Lett., 55(1082), 1985.

[49] Sauer T., Yorke J.A., and Casdagli M., *Embedology*, Journal of Statistical Physics, 65(3 - 4):579–616, 1991.

[50] Schreiber T., *Efficient neighbor searching in nonlinear time series analysis*, Int. J. Bifurcation and Chaos, 5, 1995.

[51] Schuster H.G., *Deterministic Chaos*. Weinheim: VGH Verlagsgesellschaft, 1988.

[52] Skubalska-Rafajłowicz E., *A new method of estimation of the box-counting dimension of multivariate objects using space-filling curves*, Nonlinear Analysis, 63, 2005.

[53] Takens F., *Detecting strange attractors in turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. 1981.

[54] Tricot C., *Curves and Fractal Dimension*, Soringer-Verlag, 1995.

[55] Wong A., Wu L., *Fast estimation of fractal dimension and correlation integral on stream data*, Inf. Process. Letters, 93, 2003.

[56] Yee P., Haykin S., *A dynamic regularized radial basis function network for nonlinear*, IEEE Transaction on Signal Processing, 47 1998.

[57] Yingwei L., Sundararajan N., Saratchandran P., *A sequential learning scheme for function approximation using minimal radial basis function neural network,* Neural Computation, 9:461–478, 1997.

Rozdział 4

Odporne algorytmy uczenia sieci RBF

Andrzej Rusiecki

4.1. Wprowadzenie

Rozważając metody projektowania i uczenia sztucznych sieci neuronowych, dość często zapomina się, albo świadomie ignoruje istotny problem, jakim mogą się stać błędy pojawiające się potencjalnie w danych uczących. Próby uwzględnienia możliwości występowania błędów w danych uczących zaowocowały powstaniem tzw. odpornych algorytmów uczenia sieci neuronowych. W niniejszym rozdziale omówiono istniejące rozwiązania w dziedzinie odpornych algorytmów, oraz przedstawiono nowy odporny algorytm uczenia sieci o bazach radialnych wraz z przykładowym zastosowaniem w przetwarzaniu danych strumieniowych.

4.1.1. Błędy w danych uczących a sieci neuronowe

O ile istnienie zakłóceń w postaci niewielkiego szumu nie powoduje najczęściej błędnego działania metod opartych na SSN, o tyle dodanie nawet małej liczby danych znacznie różniących się od ogółu, skutkować może ich całkowicie nieprzewidywalnym zachowaniem. Z punktu widzenia typowego użytkownika, który oczekuje, że sieć neuronowa będzie dla niego jedynie narzędziem ułatwiającym wykonanie założonego zadania, ma to stosunkowo duże znacznie, szczególnie w sytuacji, gdy sieć ma zostać użyta do znalezienia pewnych prawidłowości w zbiorze danych. Gdy owe dane są błędne, również wszelkie zaobserwowane w nich prawidłowości mogą okazać się nieprawdziwe.

Oczywiście można założyć, że w wielu sytuacjach obserwacja odbiegająca znacznie od ogółu powinna zwrócić uwagę, łatwo jednak zauważyć, że w przypadku dużej liczby danych, jak i np. pomiarów przetwarzanych bez pośrednictwa człowieka, nawet wyraźne błędy mogą pozostać niezauważone. Ponadto często trudno przewidzieć, jaki rozkład powinna mieć mierzona wielkość, w związku z czym wszystkie jej wartości muszą być wstępnie brane pod uwagę jako prawidłowe.

Aby odpowiedzieć na pytanie, czy problem ten ma jakieś praktyczne znaczenie, wystarczy uświadomić sobie, że w naukach przyrodniczych, w przypadku bez-

pośrednio mierzalnych danych pomiarowych, zawartość błędów grubych (a więc znacznie przekraczających typowe wartości) waha się zwykle w granicach kilku procent, jeśli jednak weźmie się pod uwagę bardziej opisowe dziedziny wiedzy, np. medycynę, okazuje się, że liczba danych odstających od ogółu przekracza nierzadko 10% [6], sięgając czasami 20% wszystkich obserwacji.

Na gruncie statystyki rozwija się już od dłuższego czasu nurt tzw. statystyki odpornej (ang. *Robust Statistics*) [6], [9], [13], [14] zajmującej się opisywaniem i niwelowaniem wpływu danych odstających i błędów grubych na rezultaty działania metod statystycznych. Zanim zajmiemy się możliwościami wykorzystania niektórych z jego osiągnięć w dziedzinie uczenia sieci neuronowych, warto odpowiedzieć sobie na pytanie, co rozumiemy przez pojęcie błędu grubego i danej odstającej.

4.1.2. Dane odstające od ogółu i błędy grube

Wyobraźmy sobie, że podczas akwizycji danych dotyczących kontroli jakości prostego produktu, np. średnicy nawierconych w drewnie otworów, otrzymujemy nagle odczyt różniący się od pozostałych o rząd wielkości (np. 10 cm zamiast 1 cm). Mamy wtedy do czynienia z czymś, co określa się zwykle angielskim słowem *outlier*, a co, w większości przypadków, tłumaczy się jako daną odstającą (w domyśle „od reszty danych"). W tym prostym przypadku, gdy nie staramy się konstruować matematycznego modelu na podstawie mierzonych wartości, dana taka, choć łatwa w identyfikacji, mimo wszystko wymaga uwagi, gdyż oznaczać może błąd pomiarowy lub poważną wadę produktu. Znacznie groźniejsza może być sytuacja, w której – bazując na pomiarach – chcemy określić przeciętną wielkość otworu; owa obserwacja może dość wyraźnie zakłócić wynik końcowy.

Wzorując się na [7], daną odstającą nazwiemy każdą obserwację, która odróżnia się od większości danych. Podobnie brzmi inna definicja: obserwacja, nie przystająca do obrazu tworzonego przez większość danych [6]. Widać więc, że pojęcie danych odstających, pomimo iż intuicyjnie zrozumiałe, trudno jest sprecyzować. Warto wspomnieć, że dane odstające stosunkowo często myli się, w potocznym rozumieniu, z danymi błędnymi, podczas gdy nierzadko okazują się one dostarczać najbardziej istotnych informacji. Jednak, w typowych przypadkach dane odstające mogą być rzeczywiście tożsame z błędami grubymi, będąc następstwem błędów pomiarowych lub pomyłek związanych z archiwizacją i przetwarzaniem danych.

Wyróżnić można dwa podstawowe sposoby pomagające w uwzględnieniu wpływu danych odstających [6]: po pierwsze, można w jakiś sposób zidentyfikować dane, które potencjalnie mogą być błędami i następnie przetwarzać je oddzielnie w stosunku do reszty, druga możliwość zaś to stosowanie określonych procedur eliminujących wpływ danych mogących być błędami, bez zatracania całości nie-

sionych przez nie informacji. Drugi z wymienionych sposobów, mimo że trudniejszy w realizacji, umożliwia przeprowadzenie uodporniania algorytmów uczenia sieci neuronowych.

4.1.3. Modele błędów w danych

W publikacjach dotyczących odpornych algorytmów uczenia sieci [1], [2], [10] stosowane jest najczęściej modelowanie błędów za pomocą sumy rozkładów normalnych o zerowej wartości oczekiwanej, a więc tzw. *gross error model*. Ponadto, w pracy [14] zaproponowano modelowanie błędów w podobny sposób, lecz z użyciem rozkładu niesymetrycznego, natomiast w pracy [16] model błędów, w którym mogą się one pojawiać również w wektorze wejściowym sieci czy też, ogólniej mówiąc, modelowanej relacji.

Model błędów grubych – *gross error model*

Do danych czystych, (dokładniej rzecz biorąc do wektora y) dodawane są tu zakłócenia generowane za pomocą dwóch składowych następująco:

$$F = (1 - \delta)G + \delta H, \tag{4.1}$$

gdzie F jest rozkładem błędów w danych, natomiast G i H mają rozkłady normalne o zerowej wartości oczekiwanej $N(0, \sigma_G)$ i $N(0, \sigma_H)$, przy czym $\sigma_H \gg \sigma_G$. Pierwszy ze składników to rozkład o stosunkowo niewielkiej wariancji, mający reprezentować typowy biały szum, natomiast drugi z nich ma wariancję odpowiednio dużą, by symulować występowanie w danych błędów grubych. Pojawiają się one wśród zakłóceń z założonymi prawdopodobieństwami, odpowiednio $(1-\delta)$ oraz δ.

Błędy o niesymetrycznym rozkładzie

Zakłócenia generowane są jako

$$F = (1 - \delta)G + \delta(H_1 + H_2 + H_3 + H_4). \tag{4.2}$$

Podobnie, jak wcześniej, rozkład G modeluje biały szum, natomiast, drugi z nich staje się bardziej skomplikowany, będąc złożeniem kilku dodatkowych rozkładów. Dla uproszczenia przyjęto, iż $H_i \sim N(m_i, \sigma_i)$, przy czym oba parametry rozkładów są tu z założenia niezerowe i różne dla poszczególnych H_i. Drugi z rozkładów można również zapisać jako rozkład normalny z odpowiednimi parametrami, ale dany zapis, zaproponowany w [14], podkreśla, że zakłócenia te generowane mogą być z różnych źródeł.

Błędy w wektorze wejściowym

Błędy w wektorze wejściowym, czyli tzw. *leverage points* [15], wprowadzić można zgodnie z modelem opisanym jako *gross error model*, z tą różnicą, że zakłócenia dodawane są do wartości podawanych w procesie uczenia na wejście sieci.

4.2. Odporne algorytmy uczenia

4.2.1. Wprowadzenie do odpornych algorytmów uczenia

Większość z istniejących odpornych algorytmów uczenia sieci neuronowych opiera się na wprowadzeniu nowej, różnej od typowo stosowanej kwadratowej, funkcji błędu z zachowaniem wszelkich innych cech danego algorytmu uczenia. Użycie odpowiednio skonstruowanego kryterium powoduje praktyczne wykluczenie z wpływu na proces uczenia danych powodujących największe błędy, a więc podejrzanych o bycie błędami grubymi. Odporne metody uczenia przeznaczone były początkowo jedynie do trenowania sieci sigmoidalnych, które, jak stwierdzono w [11], są z natury swej bardziej odporne niż sieci o bazach radialnych. Intuicyjnie można to uzasadnić, pamiętając, że podczas gdy w sieciach sigmoidalnych dokonuje się podziału przestrzeni pewnymi hiperpłaszczyznami, w sieciach RBF mamy do czynienia z rozciąganiem hiperpłaszczyzn wokół punktów z ciągu uczącego, tak więc, niejako automatycznie, wpływ błędów grubych może być potencjalnie większy.

Jak się jednak okazuje, można przenieść rozwiązania odpornych algorytmów uczenia bezpośrednio na grunt sieci o radialnych funkcjach bazowych. Przykładem jest tu praca [4], w której przeprowadzono próbę uodpornienia prostej sieci RBF w sposób identyczny z wcześniejszym, chronologicznie rzecz ujmując, odpornym algorytmie RBP (ang. *Robust Backpropagation*) [1] przeznaczonym do uczenia sieci o sigmoidalnych funkcjach aktywacji neuronów. Ponadto w pracy [3] powtórzono rozumowanie z [2], proponując algorytm analogiczny do opisanego dalej ARBP (ang. *Annealing Robust Backpropagation*), lecz przeznaczony do uczenia sieci z radialnymi funkcjami bazowymi.

4.2.2. Odporność na zakłócenia a dobór funkcji błędu

Prześledźmy teraz pokrótce rozumowanie leżące u podstaw odpornych algorytmów uczenia sieci, wykazujące, że odpowiednio dobrana funkcja błędu może pomóc w niwelowaniu wpływu błędów grubych na proces uczenia. Przyjmijmy dla prostoty zapisu, że sieć ma tylko jedno wyjście. W najbardziej ogólnym przy-

padku funkcja błędów może być również zależna od parametrów sieci [8], jednak najczęściej ma ona następującą postać:

$$E = \frac{1}{N}\sum_{i=1}^{N}\rho\left(r_i\right), \tag{4.3}$$

gdzie $\rho(r_i)$ oznacza funkcję kary (ang. *loss function*) [6], zwykle ciągłą i symetryczną, r_i jest błędem dla i-tego obrazu uczącego, natomiast N liczbą elementów ciągu uczącego. W najbardziej typowym przypadku za funkcję strat przyjmuje się funkcję kwadratową

$$\rho(r_i) = {r_i}^2. \tag{4.4}$$

Wtedy minimalizowane kryterium ma postać

$$E = \frac{1}{N}\sum_{i=1}^{N}{r_i}^2, \tag{4.5}$$

czyli jest równoznaczne z błędem średniokwadratowym.

Przyjęcie funkcji kwadratowej jako minimalizowanego kryterium błędów ma swoje uzasadnienie, które pokazać można, korzystając z metody największej wiarogodności [2], [9]. W tym celu załóżmy, że błędy pomiarowe r_i są niezależne i generowane według pewnego rozkładu $f(r_i)$. Możemy więc, dla ciągłego rozkładu, obliczyć prawdopodobieństwo modelu, jako iloczyn poszczególnych prawdopodobieństw

$$L = \prod_{i=1}^{N} f(r_i)\Delta, \tag{4.6}$$

przy czym $f(r_i)\Delta$ oznacza prawdopodobieństwo r_i w sąsiedztwie Δ dla ciągłej dystrybuanty. Maksymalizacja prawdopodobieństwa może zostać zapisana jako maksymalizacja jego logarytmu lub alternatywnie minimalizacja

$$\min_{\mathbf{w}} \sum_{i=1}^{N}(-\log f(r_i)) - N\log\Delta. \tag{4.7}$$

Minimalizacja odbywa się tu po wektorze parametrów $\mathbf{w}$, który w tym konkretnym przypadku może być traktowany jako wektor parametrów (wag) sieci. Ponieważ N oraz Δ to stałe, otrzymujemy

$$\min_{\mathbf{w}} \sum_{i=1}^{N}(-\log f(r_i)). \tag{4.8}$$

Dla typowego założenia, mówiącego, że błędy generowane są według rozkładu normalnego danego jako

$$f(r) = \frac{1}{\sqrt{2\pi}} \exp(-1/2r^2), \tag{4.9}$$

po podstawieniu $f(r)$ do równania (4.8) otrzymujemy

$$\min_{\mathbf{w}} \left(\sum_{i=1}^{N} \frac{1}{2} {r_i}^2 + \frac{N}{2} \log(2\pi) \right) \tag{4.10}$$

Zaniedbując stały człon, dostajemy w rezultacie właśnie minimalizację błędu kwadratowego

$$\min_{\mathbf{w}} \sum_{i=1}^{N} r_i^2. \tag{4.11}$$

Miarą oddziaływania błędów na proces uczenia jest tzw. funkcja wpływu (ang. *influence function*) [6], będąca pochodną stosowanej funkcji kary/strat:

$$\Psi(r_i) = \frac{\partial \rho(r_i)}{\partial r_i}. \tag{4.12}$$

Można ją traktować jako szczególny przypadek (a tak naprawdę rozszerzenie na teorię uczenia sieci) tak samo nazwanej funkcji używanej często w literaturze statystycznej [6], [9], w szczególności w rozważaniach dotyczących odpornych estymatorów.

Gdy policzyć funkcję wpływu dla kryterium kwadratowego, ma ona postać liniową:

$$\Psi(r_i) = 2r_i. \tag{4.13}$$

Widać więc, że błędy o największych wartościach będą miały największy wpływ na uczenie sieci z tak dobraną funkcją kryterialną.

4.3. Odporne algorytmy uczenia sieci sigmoidalnych

Jak już wspomniano, uodparnianie algorytmów uczenia sieci o sigmoidalnych funkcjach aktywacji odbywa się zazwyczaj za pomocą zmodyfikowania funkcji błędu minimalizowanej w procesie uczenia. Omówione zostaną teraz w skrócie rozwiązania znane z literatury.

4.3.1. Odporny algorytm z kryterium LMLS

W pracy Liano [10] do skonstruowania odpornego kryterium błędu zaproponowano funkcję kary nazwaną LMLS (ang. *Least Mean Log Squares*) następującej postaci:

$$\rho(r_i) = \log(1 + \frac{1}{2}{r_i}^2) \tag{4.14}$$

Wynika to stąd, że w przypadku przyjęcia ρ jak w danej zależności, funkcja wpływu jest ograniczona i można ją zapisać jako

$$\Psi(r_i) = \frac{r_i}{1 + \frac{1}{2}{r_i}^2}. \tag{4.15}$$

Dla niewielkich wartości residuów jest ona praktycznie rzecz biorąc liniowa, podczas, gdy wraz z ich wzrostem zaczyna asymptotycznie dążyć do zera. W rezultacie

$$\lim_{r\to\infty} \frac{r_i}{1 + \frac{1}{2}{r_i}^2} = 0. \tag{4.16}$$

Jak więc widać największe błędy nie mają praktycznego wpływu na wartość funkcji celu, a więc i na proces uczenia.

4.3.2. Odporny algorytm propagacji wstecznej RBP

W algorytmie tym [1] Chen *i in.* zaproponowali przyjęcie zmiennej w czasie funkcji kary bazującej na M-estymatorze opracowanym przez Hampela [6], a wykorzystującym tangens hiperboliczny tgh. Odporne M-estymatory stanowią klasę uogólnioną z estymatorów największej wiarogodności, przeznaczoną do sytuacji, w której rozkład błędu jest nieznany. Powinny one zachowywać się stabilnie dla danych czystych lub zakłócanych niewielkim białym szumem, a równocześnie zapewniać odporność procedury na występowanie błędów grubych.

W algorytmie RBP funkcja wpływu oparta na estymatorze Hampela [6] przedstawia się więc następująco:

$$\Psi_t(r_i) = \begin{cases} r & \text{dla } |r| \leqslant a(t), \\ c_1 \operatorname{tgh}(c_2(b(t) - |r|))\operatorname{sign}(r) & \text{dla } a(t) < |r| \leqslant b(t), \\ 0 & \text{dla } |r| > b(t). \end{cases} \tag{4.17}$$

Wyrażenia $a(t)$ oraz $b(t)$ oznaczają zmienne w czasie (a więc zależne od bieżącej epoki t) punkty odcięcia funkcji, c_1 i c_2 zaś pewne stałe.

Strategia zmian punktów odcięcia jest następująca: na początku zakłada się maksymalną liczbę błędów q, które mogą pojawić się w danych. Następnie wybiera się najmniejsze $(1-q)N$ bezwzględne residua $|r(t)|_{1:N} \leqslant |r(t)|_{2:N} \leqslant \ldots \leqslant$

$|r(t)|_{(1-q)N:N}$, po czym za pomocą metody bootstrap szacuje się przedział ufności spełniający zależność

$$Prob\left\{l_{(1-q)}(t) < |r(t)|_{(1-q)n:n} < u_{(1-q)}(t)\right\} = 0,95. \tag{4.18}$$

Za a i b podstawia się kolejno $l_{(1-q)}(t)$ i $u_{(1-q)}(t)$. Wartości te aktualizowane są co pewną, ustaloną liczbę epok procesu uczenia. Dla uproszczenia założyć można mniej skomplikowaną postać funkcji kary, mającą tylko jeden punkt odcięcia. I tak, dla funkcji Cauchy'ego otrzymujemy

$$\rho(r_i, \beta) = \frac{\beta}{2} \log\left(1 + \frac{r_i^2}{\beta}\right), \tag{4.19}$$

przy czym parametr β przyjmowany jest jako

$$\beta(t) = |r(t)|_{(1-q)N:N}. \tag{4.20}$$

4.3.3. Odporny algorytm propagacji wstecznej z wyżarzaniem ARBP

Funkcja strat została tu przyjęta jak w zależności (4.19). Autorzy pracy [2] zaproponowali jednak, aby zmiany parametru β odbywały się zgodnie z zadanym schematem wyżarzania. Umożliwia to działanie algorytmu, w którym zbędne staje się początkowe założenie dotyczące liczby błędów grubych zawartych w danych uczących. W pierwszej fazie uczenia minimalizowane kryterium powinno być bliskie błędowi średniokwadratowemu, by po pewnej liczbie iteracji zacząć eliminować dane powodujące największe błędy. Aby osiągnąć taki rezultat, punkt odcięcia musi zmieniać się w czasie i zaczynając od stosunkowo dużej wartości, stopniowo dążyć do zera. Wykazano eksperymentalnie, że najbardziej efektywny jest schemat wyżarzania postaci $\beta(t) = k/t$, gdzie k oznacza pewną stałą, przyjętą jako $k = 10$, gdyż taka wartość sprawdzała się najlepiej dla testowych przypadków.

4.3.4. TAO – odporny algorytm propagacji wstecznej

Autorzy algorytmu TAO, Pernia-Espinoza *i in.* [14] również skupili się na zwiększeniu odporności procesu uczenia sieci, na błędy pojawiające się w danych uczących przez wprowadzenie nowej funkcji kryterialnej. Funkcja ta obliczana jest w dość skomplikowany sposób, który nie zostanie tu zaprezentowany, a do jej skonstruowania posłużono się ideą τ-estymatorów zaproponowanych przez Yohai i Zamara w 1988 r.[18].

4.3.5. Odporny algorytm LTS

Zaproponowany w pracy [16] odporny algorytm opiera się z kolei na odpornym estymatorze zwanym estymatorem najmniejszych przycinanych kwadratów, w którym część błędów nie jest w ogóle brana pod uwagę.

Odporne kryterium błędu LTS bazujące na estymatorze najmniejszych przyciętych kwadratów definiowane jest następująco:

$$E_{LTS} = \sum_{i=1}^{h} (r^2)_{i:N}. \tag{4.21}$$

W tym przypadku $(r^2)_{1:N} \leqslant \cdots \leqslant (r^2)_{N:N}$ to podniesione do kwadratu residua w postaci

$$r_i^2 = \{\sum_{v=1}^{m} |(\hat{y}_{iv} - y_{iv})|\}^2, \tag{4.22}$$

gdzie $\hat{y}_{iv}$, oraz y_{iv} oznaczają kolejno wartość wyjścia v i zadaną wartość na wyjściu v dla obrazu uczącego i, a sumowanie odbywa się po wszystkich m wyjściach sieci. Od wielkości stałej przycinania h zależy liczba obrazów uczących, które będą traktowane jako punkty odstające, dlatego sposób jej doboru ma istotne znaczenie dla właściwości nowego kryterium.

4.4. Odporne algorytmy uczenia sieci o bazach radialnych

W przypadku sieci o radialnych funkcjach bazowych odporne algorytmy również tworzone są według jednego schematu. Pierwszy etap, w którym wyznaczana jest liczba neuronów (a więc centrów) i parametry funkcji radialnych, odbywa się bez uwzględniania wpływu błędów grubych. Dopiero w drugim etapie, podczas douczania sieci, wprowadzane są modyfikacje uodparniające. Polegają one, podobnie jak w przypadku algorytmów uczenia sieci sigmoidalnych, na zastosowaniu nowego kryterium minimalizowanego w procesie uczenia.

4.4.1. Zastosowanie algorytmu RBP do uczenia sieci RBF

Przeniesienie idei odpornego algorytmu uczenia RBP znaleźć można w pracy [4], której autor w bezpośredni sposób zastosował funkcję kryterialną daną zależnością (4.19), zaproponowaną w [1]. W algorytmie tym początkowe parametry i architektura sieci RBF ustalane są metodą opartą na rozkładzie według wartości szczególnych SVD (ang. *singular value decomposition*), następnie oblicza się, na podstawie założonej apriorycznie maksymalnej liczby błędów w danych,

punkt odcięcia funkcji. Dalej następuje douczenie sieci, a więc minimalizacja nowej funkcji błędu względem wektora wag wyjściowych, przeprowadzone metodą gradientów sprzężonych.

4.4.2. Odporna sieć ARRBFN

W pracy [3] przedstawiono odporną sieć o bazach radialnych ARRBFN (ang. *Annealing Robust Radial Basis Function Network*). Jak sama nazwa wskazuje, autorzy zaproponowali przeniesienie swojej metody uczenia sieci sigmoidalnych [2] na sieci RBF. Początkową strukturę sieci, wagi i parametry funkcji bazowych uzyskiwane są na podstawie metody wektorów wspierających SVR (ang. *Support Vector Regression*) [17]. Dopiero potem rozpoczyna się douczanie sieci przez proces minimalizacji funkcji błędu, danej zależnością (4.19), przy czym parametr β zmieniany jest w sposób identyczny z opisanym podczas omawiania metody ARBP.

4.5. Szybki odporny algorytm uczenia sieci o radialnych funkcjach bazowych

Dalej zaprezentowano prosty, a zarazem szybki i efektywny algorytm pomagający uodpornić sieci o radialnych funkcjach bazowych. Może on być szczególnie polecany do zastosowania w zadaniach wymagających szybkiego przeprowadzania procesu uczenia. Algorytm ten opiera się na funkcji błędu LMLS zaproponowanej w pracy [10], w której funkcja kary dana jest zależnością (4.14).

Podobnie, jak we wspomnianych odpornych algorytmach uczenia sieci RBF, założymy, że początkowa struktura, a więc liczba neuronów, wagi i parametry funkcji bazowych uzyskiwane są jedną ze standardowych metod, bez uwzględniania możliwości pojawienia się błędów w danych uczących. Istotny jest tu drugi etap uczenia, a więc adaptacja wag sieci, mająca na celu lepsze odzwierciedlenia danych uczących.

Zapiszmy więc, jak dokładnie przedstawia się całkowity błąd sieci obliczany w konkretnej epoce, po zastosowaniu uodparniającej modyfikacji:

$$E(\mathbf{w}) = \sum_{k=1}^{N} \sum_{i=1}^{m} \log(1 + \frac{1}{2} r_{ki}^2(\mathbf{w})), \tag{4.23}$$

gdzie $r_{ki} = (\hat{y}_{ki}(\mathbf{w}) - y_{ki})$ oznacza błąd popełniany przez i-te wyjście sieci dla k-tego obrazu uczącego, m jest liczbą wyjść sieci, N liczbą obrazów uczących. Dla tak sformułowanej funkcji błędu można teraz przeprowadzić uczenie, np.

jedną z metod gradientowych, możliwych do zastosowania podczas trenowania sieci jednokierunkowych.

Autorzy cytowanych odpornych metod uczenia proponowali douczanie sieci algorytmem największego spadku [3] lub gradientów sprzężonych [4]. To pierwsze rozwiązanie uznać można raczej za historyczne w ujęciu praktycznych zastosowań, biorąc pod uwagę, że istnieją inne, zdecydowanie bardziej wydajne metody. Algorytm gradientów sprzężonych jest jedną z popularniejszych, jednak w niniejszych rozważaniach zaprezentowano nowy algorytm przeznaczony do użycia z konkretną funkcją błędu. Procedurę optymalizacyjną wyprowadzić można tu podobnie jak dla często stosowanego algorytmu Levenberga–Marquardta [5], [12]. W algorytmie tym, dzięki odpowiedniemu oszacowaniu macierzy hesjanu i zastosowaniu zmiennego w czasie czynnika regularyzacyjnego początkowo przeprowadzana jest minimalizacja funkcji celu metodą największego spadku, natomiast w pobliżu rozwiązania hesjan aproksymowany jest jedynie za pomocą rozwinięcia pierwszego rzędu, co prowadzi do przejścia do algorytmu Gaussa–Newtona o kwadratowej zbieżności i niezmienionym nakładzie obliczeniowym. Podejście takie znaczenie przyspiesza proces uczenia sieci neuronowej. Niestety, wszelkie przybliżenia opierają się tam na założeniu kwadratowej funkcji błędu, nie da się więc przenieść ich do odpornego algorytmu uczenia.

Dla funkcji błędu danej zależnością (4.23) należy więc spróbować znaleźć inne oszacowania. We wprowadzonych wcześniej oznaczeniach, możemy zapisać jakobian funkcji celu jako

$$J(\mathbf{w}) = \begin{bmatrix} \frac{\partial r_{11}}{\partial w_1} & \frac{\partial r_{11}}{\partial w_2} & \cdots & \frac{\partial r_{11}}{\partial w_n} \\ \frac{\partial r_{21}}{\partial w_1} & \frac{\partial r_{21}}{\partial w_2} & \cdots & \frac{\partial r_{21}}{\partial w_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_{N1}}{\partial w_1} & \frac{\partial r_{N1}}{\partial w_2} & \cdots & \frac{\partial r_{N1}}{\partial w_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_{1m}}{\partial w_1} & \frac{\partial r_{1m}}{\partial w_2} & \cdots & \frac{\partial r_{1m}}{\partial w_n} \\ \frac{\partial r_{2m}}{\partial w_1} & \frac{\partial r_{2m}}{\partial w_2} & \cdots & \frac{\partial r_{2m}}{\partial w_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_{Nm}}{\partial w_1} & \frac{\partial r_{Nm}}{\partial w_2} & \cdots & \frac{\partial r_{Nm}}{\partial w_n} \end{bmatrix}, \tag{4.24}$$

gdzie

$$r(\mathbf{w}) = \begin{bmatrix} r_{11} & \cdots & r_{N1} & r_{12} & \cdots & r_{N2} & \cdots & r_{1m} & \cdots & r_{Nm} \end{bmatrix}^T, \tag{4.25}$$

a n jest liczbą wag w sieci.

Wprowadźmy teraz pomocniczy wektor p

$$p(\mathbf{w}) = \left[\begin{array}{cccc} \frac{r_{11}}{1+\frac{1}{2}{r_{11}}^2} & \frac{r_{21}}{1+\frac{1}{2}{r_{21}}^2} & \frac{r_{31}}{1+\frac{1}{2}{r_{31}}^2} & \cdots & \frac{r_{Nm}}{1+\frac{1}{2}{r_{Nm}}^2} \end{array} \right]^T. \tag{4.26}$$

Jak widać, powstał on po nieskomplikowanych algebraicznych przekształceniach dokonanych na wektorze residuów. Biorąc pod uwagę, że jakobian zdefiniowany jest zależnością (4.24), możemy zapisać dokładnie gradient funkcji celu

$$\nabla E(\mathbf{w}) = J(\mathbf{w})^T p(\mathbf{w}). \tag{4.27}$$

Zdefiniujmy teraz drugi wektor pomocniczy $q(\mathbf{w})$, wykonując operacje arytmetyczne na elementach wektora r postaci

$$\left[\begin{array}{cccc} \frac{1}{1+\frac{1}{2}{r_{11}}^2} - \frac{{r_{11}}^2}{(1+\frac{1}{2}{r_{11}}^2)^2} & \frac{1}{1+\frac{1}{2}{r_{21}}^2} - \frac{{r_{21}}^2}{(1+\frac{1}{2}{r_{21}}^2)^2} & \cdots & \frac{1}{1+\frac{1}{2}{r_{Nm}}^2} - \frac{{r_{Nm}}^2}{(1+\frac{1}{2}{r_{Nm}}^2)^2} \end{array} \right]^T.$$

Zapiszmy jeszcze dodatkową macierz JQ, która powstała przez pomnożenie każdego wiersza jakobianu przez odpowiadający mu element $q(\vec{w})$

$$JQ(\mathbf{w}) = \left[\begin{array}{cccc} \frac{\partial r_{11}}{\partial w_1} q_{11} & \frac{\partial r_{11}}{\partial w_2} q_{11} & \cdots & \frac{\partial r_{11}}{\partial w_n} q_{11} \\ \frac{\partial r_{21}}{\partial w_1} q_{22} & \frac{\partial r_{21}}{\partial w_2} q_{22} & \cdots & \frac{\partial r_{21}}{\partial w_n} q_{22} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_{N1}}{\partial w_1} q_{N1} & \frac{\partial r_{N1}}{\partial w_2} q_{N1} & \cdots & \frac{\partial r_{N1}}{\partial w_n} q_{N1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_{1m}}{\partial w_1} q_{1m} & \frac{\partial r_{1m}}{\partial w_2} q_{1m} & \cdots & \frac{\partial r_{1m}}{\partial w_n} q_{1m} \\ \frac{\partial r_{2m}}{\partial w_1} q_{2m} & \frac{\partial r_{2m}}{\partial w_2} q_{2m} & \cdots & \frac{\partial r_{2m}}{\partial w_n} q_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_{Nm}}{\partial w_1} q_{Nm} & \frac{\partial r_{Nm}}{\partial w_2} q_{Nm} & \cdots & \frac{\partial r_{Nm}}{\partial w_n} q_{Nm} \end{array} \right]. \tag{4.28}$$

Mając zdefiniowane dane macierze, możemy już zapisać zależność na macierze hesjanu, która dana jest w tym przypadku równaniem następującym:

$$H(\mathbf{w}) = J(\mathbf{w})^T JQ(\mathbf{w}) + S(\mathbf{w}). \tag{4.29}$$

Tym sposobem otrzymaliśmy prostą obliczeniowo przybliżoną zależność umożliwiającą obliczenie hesjanu. Przybliżenie polega tu na pominięciu składnika S, który zależy bezpośrednio od drugich pochodnych i zapisany może być jako

$$S(\mathbf{w}) = \sum_{k=1}^{N} \sum_{i=1}^{m} p_{ki}(\mathbf{w}) \nabla^2 r_{ki}(\mathbf{w}) \tag{4.30}$$

Dane uproszczenie znacznie ułatwia i przyspiesza obliczenia. Analogicznie do algorytmu Leveberga–Marquardta do zależności opisującej hesjan wprowadzimy czynnik regularyzacyjny u, otrzymując ostateczną postać równania

$$H(\mathbf{w}) = J(\mathbf{w})^T JQ(\mathbf{w}) + u\mathbf{1}, \tag{4.31}$$

gdzie $\mathbf{1}$ oznacza macierz jednostkową. Zmiany czynnika $u(t)$ (t oznacza epokę) w procesie uczenia przedstawiają się następująco: w danej epoce jest on zwiększany według schematu $\eta u(t)$ aż do momentu uzyskania poprawy wartości funkcji celu. Następnie, po wykonaniu kroku uczenia, jest on zmniejszany jak $\beta u(t)$ w głównym algorytmie. Stałe η i β zostały przyjęte jak w oryginalnym algorytmie Levenberga–Marquardta.

Możemy więc już zapisać, w jaki sposób aktualizowane będą wagi sieci w kolejnych epokach

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \Big[J(\mathbf{w}_t)^T JQ(\mathbf{w}_t) + u(t)\mathbf{1}\Big]^{-1} \nabla E(\mathbf{w}_t). \tag{4.32}$$

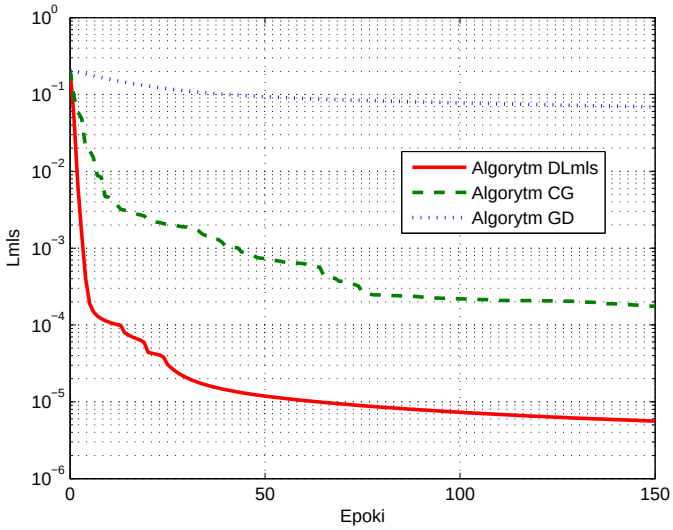
Otrzymaliśmy więc nowy sposób uczenia sieci neuronowej jednokierunkowej (nie tylko o radialnych funkcjach bazowych) dostosowany do odpornego kryterium błędu.

Na rysunkach 4.1 i 4.2 pokazano przykładowy przebieg procesu uczenia algorytmami największego spadku, gradientów sprzężonych, oraz opisaną metodą (oznaczoną jako DLmls), na danych czystych i zawierających zakłócenia. Przedstawione rysunki dotyczą sieci jednokierunkowej, dwuwarstwowej o neuronach z sigmoidalną funkcją aktywacji uczonej zadania aproksymacji.

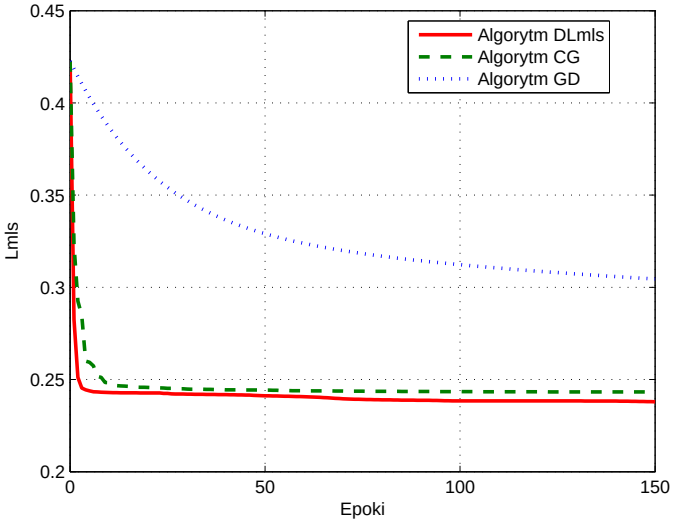
4.6. Przetwarzanie danych strumieniowych w zadaniu sterowania

Jak już wspomniano, zaletą opisanego odpornego algorytmu uczenia, w porównaniu z innymi algorytmami mającymi zapewniać odporność na błędy w danych, jest szybkość jego działania. Z tego powodu uzasadnione wydaje się użycie go w zadaniu, gdzie szybkość przetwarzania danych jest parametrem kluczowym. Aby jednak rezultaty symulacji dało się w sposób nieskomplikowany zaprezentować i porównać, skupimy się na sytuacji, w której przetwarzane dane nie mają wielu wymiarów, choć napływają w sposób ciągły. Postulat taki spełnia zagadnienie wykorzystania sieci neuronowej w sterowaniu z modelem odniesienia (ang. *model reference control*) lub sterowaniu predykcyjnym (ang. *predictive control*).

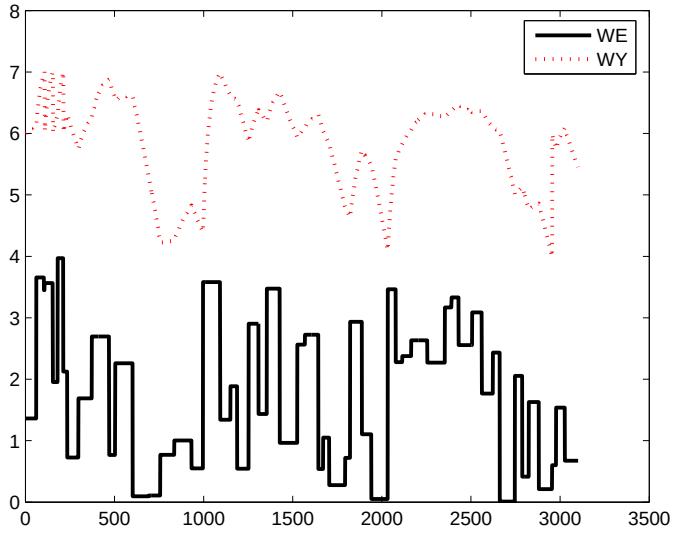
Sieć neuronowa użyta być tu może do identyfikacji sterowanego procesu. Na podstawie danego pobudzenia – oraz zarówno kilku poprzednich, jak i aktualnych wartości wejść i wyjść systemu – sieć taka powinna przewidzieć jego zachowanie



Rysunek 4.1. Porównanie przebiegu uczenia z kryterium LMLS dla algorytmu największego spadku (GD), gradientów sprzężonych (CG) oraz szybkiego algorytmu DLMLS (dane uczące bez błędów grubych)



Rysunek 4.2. Porównanie przebiegu uczenia z kryterium LMLS dla algorytmu największego spadku (GD), gradientów sprzężonych (CG) oraz szybkiego algorytmu DLMLS (dane z błędami)



Rysunek 4.3. Przeskalowane sygnały pobudzenia i wyjścia sterowanego układu

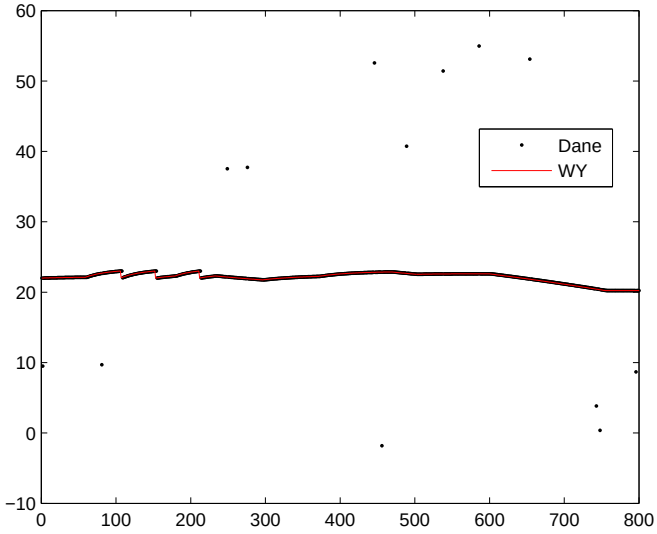
w następnej chwili. Służyć może ona więc jako model umożliwiający dostosowanie sterowania do przewidywanych zachowań systemu.

4.6.1. Zadanie testowe

Sterowanym procesem był tu tzw. chemiczny reaktor przepływowy z mieszaniem CSTR (ang. *Continuous Stirred Tank Reactor*), w którym zadaniem układu sterującego jest utrzymywanie stałego stężenia produktu reakcji przez odpowiednie dozowanie ilości jednego z dwóch substratów.

Z naszego punktu widzenia istotna jest część zadania, w której wykorzystana być może sieć RBF. Na postawie danych pomiarowych sieć taka uczona jest przewidywania reakcji układu na sterowanie w danych warunkach. Aby rozpocząć tworzenie sieciowego modelu procesu, niezbędny jest zbiór danych uczących składający się z podawanych na wejście układu sygnałów sterowania i odpowiadający im zbiór wartości wyjściowych. Przykład reakcji układu na pobudzenie przedstawiono na rysunku 4.3.

Założono, że do predykcji wyjścia systemu, sieć RBF dysponować będzie informacją na temat wartości sygnałów pobudzenia i wyjścia zarówno aktualnych, jak i z dwóch poprzednich chwil. Innymi słowy, zadaniem sieci jest przewidzenie wyjścia systemu w następnym momencie na podstawie aktualnego i poprzedniego



Rysunek 4.4. Zakłócone dane uczące na tle prawidłowego sygnału wyjścia układu

pobudzenia oraz aktualnej i poprzedniej wartości na wyjściu. Takie postawienie problemu jednoznacznie określa więc, że sieć musi mieć cztery wejścia i jedno wyjście.

Aby zamodelować błędy pojawiające się w danych uczących, posłużono się, opisanym wcześniej, modelem błędów grubych. Przykład zakłóconych w ten sposób wartości wyjścia układu przedstawiono na rysunku 4.4. Należy zwrócić uwagę na fakt, że pojedyncza zakłócona wartość skutkuje większą liczbą zakłóceń w danych uczących i to zarówno w wektorze wejściowym, jak i wyjściowym sieci, ze względu na wykorzystanie w procesie uczenia informacji o stanie układu w kilku poprzedzających momentach. Oprócz typowych błędów grubych, mamy tu więc do czynienia ze wspomnianym już zjawiskiem tzw. punktów dźwigniowych (ang. *leverage points*).

4.6.2. Wyniki symulacji

Na rysunku 4.5 pokazano, dla porównania, wyniki działania sieci zwykłej i uodpornionej na tle uzyskanych na podstawie modelu matematycznego odpowiedzi identyfikowanego układu. Jak można zauważyć, sygnały wyjściowe przewidziane przez obie sieci odbiegają od rzeczywistych wartości na wyjściu sterowanego procesu. Sieć uczona w sposób tradycyjny dość mocno dopasowała się do

Tabela 4.1. Porównanie uśrednionych ze 100 przypadków błędów średniokwadratowych sieci douczonej odpornym algorytmem (RRBF) i uczonej bez uwzględnienia błędów grubych (RBF)

L. zakłóceń	0,01	0,007	0,005
Sieć RBF	8,0347 ± 6,9524	2,1908 ± 2,0571	0,8000 ± 0,6050
Sieć RRBF	1,8883 ± 3,0339	1,1545 ± 1,4723	0,7668 ± 0,6380

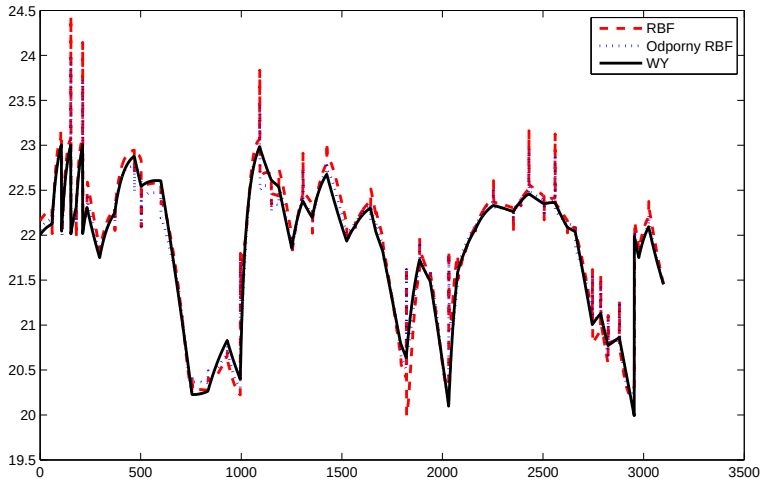
danych uczących, czego efektem są charakterystyczne skoki, a więc stosunkowo gwałtowne zmiany jej sygnału wyjściowego. Zastosowanie modyfikacji algorytmu uczenia, mającej na celu uodpornienie sieci na błędy w danych uczących, dało zdecydowanie lepsze rezultaty. Nadal widoczne są tu błędne skoki na wyjściu sieci, mają one jednak łagodniejszy charakter.

Analizując tabelę 4.1, w której zawarte zostały uśrednione dla 100 symulacji błędy popełniane przez sieci, łatwo zauważyć, że sieć uodporniona uzyskuje rezultaty lepsze niż sieć uczona w sposób tradycyjny. Dla najmniejszej liczby zakłóceń średnie błędy dla obu sieci są tego samego rzędu, przy czym sieć odporna działa nieznacznie lepiej. Dla przypadku, w którym zakłócenia stanowiły średnio 0,7% danych uczących, błąd uzyskany dla sieci odpornej jest już prawie dwukrotnie mniejszy niż dla zwykłej sieci RBF. Trzeba tu zauważyć, że odchylenia standardowe błędów są tu już zbliżone do wartości średnich. W sytuacji, gdy zakłócenia stanowią średnio 1% błędów, sieć uodporniona umożliwia osiągnięcie średniego błędu ponad czterokrotnie niższego niż sieć klasyczna. Widać jednak, że ten poziom błędów grubych jest zbyt duży dla obu sieci, gdyż rozrzut otrzymanych wyników był znaczący.

4.7. Podsumowanie

W pracy poruszony został problem błędów grubych mogących zakłócać proces uczenia i tworzenia sztucznych sieci neuronowych. To istotne zagadnienie zostało tu jedynie dość ogólnie zasygnalizowane przez przedstawienie istniejących rozwiązań problemu, dotyczących przede wszystkim sieci jednokierunkowych o sigmoidalnych funkcjach aktywacji neuronów. Jak widać, tematyka ta, w odniesieniu do sieci o radialnych funkcjach bazowych, wydaje się kryć spory potencjał co do możliwości wprowadzenia udoskonaleń mających na celu uzyskanie odporności na błędy grube pojawiające się w danych uczących.

Zaprezentowany został również nowy algorytm uczenia, który, pomimo swej prostoty, umożliwia polepszenie jakości działania sieci trenowanej w obecności dużych zakłóceń. Szczególnie godny podkreślenia jest fakt, że metoda ta może



Rysunek 4.5. Wyniki działania sieci RBF uczonej algorytmem tradycyjnym i odpornym

być stosowana już po wstępnym utworzeniu struktury sieci RBF i doborze jej parametrów. Ma ona za zadanie jedynie pewne przesunięcie wag sieci w kierunku minimum nowej funkcji kryterialnej, dyskryminującej dane powodujące największe błędy. Oznacza to, że najważniejszy etap uczenia odbywa się bez uwzględnienia możliwości pojawienia się dużych zakłóceń. Można zatem wnioskować, że dalsze możliwe do poczynienia kroki, takie jak uodpornienie owej pierwszej części algorytmu uczenia, skutkować by mogły radykalną poprawą jakości działania sieci uczonej na danych zawierających błędy grube.

Stosowanie odpornych metod uczenia sieci neuronowych, nawet w ich najmniej skomplikowanej formie, może znacząco poprawić jakość działania sieci, szczególnie w przypadku rzeczywistych zastosowań, kiedy często jakość i wiarygodność danych uczących są nieznane.

Bibliografia

[1] Chen D.S., Jain R.C., *A robust back propagation learning algorithm for function approximation*, IEEE Transactions on Neural Networks, Vol. 5, 467–479, May 1994.

[2] Chuang C., Su S., Hsiao C., *The Annealing Robust Backpropagation (ARBP) Learning Algorithm*, IEEE Transactions on Neural Networks, Vol. 11, 1067–1076, September 2000.

[3] Chuang C.C., Jeng J.T., Lin P.T., *Annealing robust radial basis function networks for function approximation with outliers*, Neurocomputing, Vol. 56, 123–139, 2004.

[4] David Sanchez V.A., *Robustization of a learning method for RBF networks*, Neurocomputing, Vol. 9, 85–94, 1995.
[5] Hagan M.T., Menhaj M.B., *Training Feedforward Networks with the Marquardt Algorithm*, IEEE Trans. on Neural Networks, Vol. 5, No. 6, 1994.
[6] Hampel F.R., Ronchetti E.M., Rousseeuw P.J., Stahel W.A., *Robust Statistics the Approach Based on Influence Functions*, John Wiley & Sons, New York, 1986.
[7] Hawkins D.M., *Identification of Outliers*, Chapman and Hall, London, 1980.
[8] Haykin S., *Neural Networks – A Comprehensive Foundation*, Second Edition, Prentice Hall, N.J., 1999.
[9] Huber P.J., *Robust Statistics*, Wiley, New York, 1981.
[10] Liano K., *Robust error measure for supervised neural network learning with outliers*, IEEE Transactions on Neural Networks, Vol. 7, 246–250, Jan. 1996.
[11] Liu J., Gader P., *Outlier Rejection with MLPs and Variants of RBF Networks*, 680–683, IEEE 2000.
[12] Marquardt D., *An algorithm for least squares estimation of non–linear parameters*, J. Soc. Ind. Appl. Math., 431–441, 1963.
[13] Olive D.J., *Applied Robust Statistics*, praca niepublikowana, 2007.
[14] Pernia–Espinoza A.V., Ordieres–Mere J.B., Martinez–de–Pison F.J., Gonzalez–Marcos A., *TAO–robust backpropagation learning algorithm*, Neural Networks, Vol.18, 191–204, 2005.
[15] Rousseeuw P.J., Leroy A.M., *Robust Regression and Outlier Detection*, Wiley, New York, 1987.
[16] Rusiecki A.L., *Robust LTS Backpropagation Learning Algorithm*, IWANN 2007, LNCS, Vol.4507, 102–109, Springer–Verlag 2007.
[17] Smola A.J., Schoelkopf B., *From regularization operators to support vector kernels*, Neural Inf. Process. Syst. 10, 343–349, 1998.
[18] Yohai V., Zamar R., *High breakdown–point estimates of regression by means of the minimization of an efficient scale*, Journal of the American Statistical Association, 83(402), 406–413, 1988.

Rozdział 5

Uczenie ortogonalnych sieci neuronowych

Krzysztof Halawa

W rozdziale tym przedstawiono ortogonalne sieci neuronowe ONN. Wymieniono część z ich istotnych zalet oraz najważniejsze wady. Opisano sposoby uczenia ONN ze szczególnym uwzględnieniem metod odpornych na błędy grube, które często stanowią poważny problem w wielu aspektach związanych z przetwarzaniem i akwizycją danych. Pokazano także algorytm szybkiego obliczania wyjść ONN mających neurony, których funkcje aktywacji należą do szeregu trygonometrycznego.

5.1. Wprowadzenie

Niewiele prac zostało poświęconych ONN. Większość z nich powstała w ostatnich dwóch dekadach. Sieci te zostały przedstawione m.in. w [18], [20], [23], [26]. ONN mają wiele ważnych zalet, wśród których można wymienić:

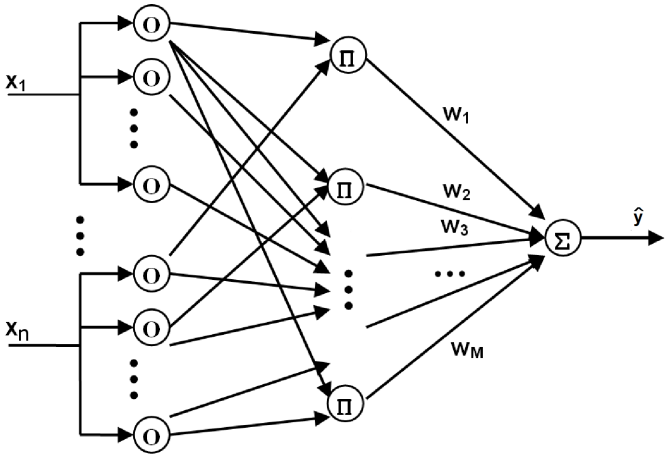
- liniową zależność wyjścia od wag,
- bardzo szybki proces uczenia spowodowany brakiem minimów lokalnych wartości oczekiwanych kilku popularnych funkcji celu (w tym także błędu średniokwadratowego),
- znana jest zależność między liczbą neuronów a liczbą wyjść i wejść sieci,
- brak problemów z rozmieszczeniem centrów, które występują w sieciach radialnych,
- znacznie łatwiejsza jest interpretacja znaczenia wartości poszczególnych wag niż w przypadku sieci sigmoidalnych,
- istnieje spora liczba algorytmów, które nadają się do zmniejszenia wpływu elementów odstających.

ONN są sieciami jednokierunkowymi.

5.2. Struktura sieci

Sygnały z wejść ONN pobudzają neurony znajdujące się w warstwie ukrytej. Niech n oznacza liczbę wejść sieci. Z i-tym wejściem ONN jest związanych M_i

neuronów w warstwie ukrytej, gdzie $i = 1, 2, \ldots, n$. Neurony te mają ortogonalne funkcje aktywacji. Mają one tylko jedno wejście. Węzły mnożące wyznaczają iloczyny wszystkich kombinacji wyjść wspomnianych neuronów. W warstwie wyjściowej znajdują się neurony liniowe. Ich liczba jest równa liczbie wyjść ONN. Neurony liniowe obliczają sumy ważone wyników otrzymanych z węzłów mnożących. Ich wagi są zmieniane podczas procesu uczenia sieci. Struktura ONN mająca kilka wejść i jedno wyjście MISO (ang. *Multiple Inputs, Single Output*) została przedstawiona na rysunku 5.1. Sieć z większą liczbą wyjść MIMO (ang. *Multiple Inputs, Multiple Output*) pokazano na rysunku 5.2.

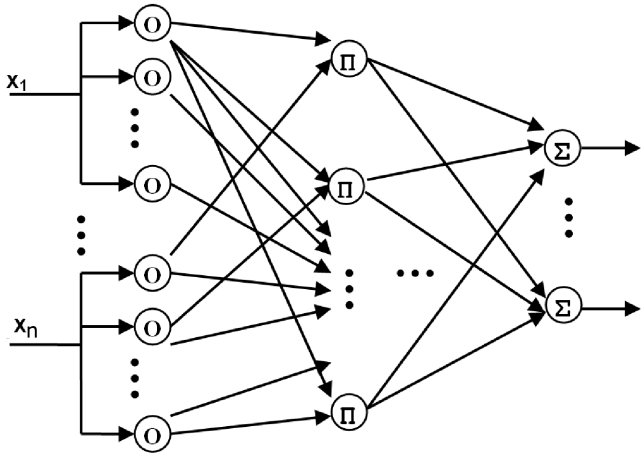


Rysunek 5.1. Struktura MISO ONN (O – neurony w warstwie ukrytej, Π – węzły mnożące, Σ – neuron liniowy)

Wartość wyjścia MISO ONN wynosi

$$\hat{y}(\mathbf{x}) = \mathbf{w}^T \left(\begin{bmatrix} f_1(x_1) \\ f_2(x_1) \\ \vdots \\ f_{M_1}(x_1) \end{bmatrix} \otimes \begin{bmatrix} f_1(x_2) \\ f_2(x_2) \\ \vdots \\ f_{M_2}(x_2) \end{bmatrix} \otimes \ldots \otimes \begin{bmatrix} f_1(x_n) \\ f_2(x_n) \\ \vdots \\ f_{M_n}(x_n) \end{bmatrix} \right), \tag{5.1}$$

gdzie $x_1, \ldots, x_n$ są wejściami sieci, $\mathbf{x} = [x_1, \ldots, x_n]^T$, $\otimes$ oznacza iloczyn Kroneckera, $f_1, f_2, \ldots$ są funkcjami aktywacji neuronów w warstwie ukrytej, $\mathbf{w} = [w_1, w_2, \ldots, w_M]^T$ jest wektorem wag liniowego neuronu, $M = \prod_{i=1}^{n} M_i$. Warto zwrócić uwagę, że elementy wektora, który otrzymamy po wyliczeniu iloczynów Kroneckera wewnątrz nawiasu okrągłego we wzorze (5.1), są równe wyjściom



Rysunek 5.2. Struktura MIMO ONN

węzłów mnożących. Można zastosować np. następujące ortogonalne funkcje aktywacji:

- szereg funkcji sinus i cosinus $c_1, c_2\sin(x), c_2\sin(2x), \ldots, c_2\cos(x), c_2\cos(2x), \ldots$, gdzie c_1 i c_2 są stałymi,
- wielomiany Legendre'a.

Sieci, których neurony w warstwie ukrytej mają funkcje aktywacji z szeregu wymienionego na pierwszej pozycji podanej listy, są nazywane fourierowskimi sieciami neuronowymi FSNN (ang. *Fourier Series Neural Network*). Przeważnie przyjmuje się $c_1 = 1/\sqrt{2\pi}$, $c_2 = 1/\sqrt{\pi}$. Wówczas wspomniane funkcje są ortonormalne. Opis właściwości wielomianów Legendre'a i szeregu funkcji sinus i cosinus znajduje się w [3]. Sygnały podawane na wejścia sieci muszą być tak przeskalowane, aby ich maksymalne wartości zawierały się w przedziale, na którym funkcje aktywacji są ortogonalne. Dla FSNN $x_1, \ldots, x_n$ muszą należeć do przedziału $[0, 2\pi)$ lub $[-\pi, \pi)$. W przypadku ONN z neuronami, których funkcje aktywacji są wielomianami Legendre'a, $x_1, \ldots, x_n \in [-1, 1]$.

5.3. Metody uczenia

W punkcie tym założono, że dla MISO ONN dysponujemy zbiorem danych uczących $\{y_k, \mathbf{x}_k\}_{k=1}^N$, gdzie $y_k = d_k + \varepsilon_k$, d_k jest wartością zadaną wyjścia sieci, gdy jej wejścia są równe elementom wektora $\mathbf{x}_k = [x_{1,k}, x_{2,k}, \ldots, x_{n,k}]^T$, ε_k są

niezależnymi zmiennymi losowymi o jednakowym rozkładzie, zerowej wartości oczekiwanej i skończonej wariancji σ^2.

Najczęściej używaną funkcją celu jest błąd średniokwadratowy

$$E_1 = \frac{1}{2}\sum_{k=1}^{N}\left(y_k - \hat{y}(\mathbf{x}_k)\right)^2, \tag{5.2}$$

gdzie $\hat{y}(\mathbf{x}_k)$ oznacza wartość wyjścia ONN, gdy wejścia sieci są równe elementom wektora $\mathbf{x}_k$.

W [26] wykazano, że wartość oczekiwana błędu średniokwadratowego MSE (ang. *Mean Squared Error*) jest n-wymiarową hiperparabolą.

Zaletą wielu algorytmów gradientowych jest możliwość ich wykorzystania do minimalizacji innych funkcji celu niż (5.2) oraz fakt, że zmiana wag może następować każdorazowo po prezentacji kolejnych par $\{y_k, \mathbf{x}_k\}$ (tzw. uczenie *on-line*). Algorytmy te nadają się do zastosowania w układach adaptacyjnych, gdzie sieć jest bezustannie trenowana. Rozwijając wybraną funkcję celu E w szereg Taylora w otoczeniu wektora $\mathbf{w}$ w kierunku $\mathbf{q} \in \mathbb{R}^M$, otrzymujemy

$$E(\mathbf{w}+\mathbf{q}) = E(\mathbf{w}) + \left(\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}\right)^T \mathbf{q} + \frac{1}{2}\mathbf{q}^T\mathbf{H}(\mathbf{w})\mathbf{q} + \ldots, \tag{5.3}$$

gdzie $\mathbf{H}$ jest hesjanem (macierzą drugich pochodnych cząstkowych) funkcji celu względem wag

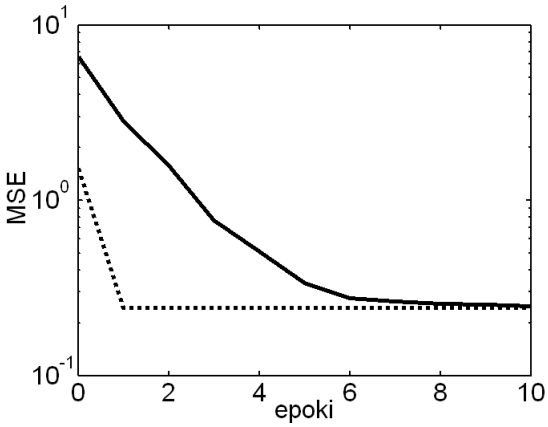
$$\mathbf{H}(\mathbf{w}) = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_M} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_M \partial w_1} & \frac{\partial^2 E}{\partial w_M \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_M^2} \end{bmatrix}.$$

Algorytmy, które wykorzystują informację zawartą zarówno w gradiencie, jak i hesjanie lub jego przybliżeniu, są nazywane algorytmami drugiego rzędu w odróżnieniu od algorytmów pierwszego rzędu, które używają tylko liniowego przybliżenia $E(\mathbf{w}+\mathbf{q})$.

Do uczenia ONN nie są stosowane algorytmy gradientowe drugiego rzędu, ponieważ wyznacznik z hesjanu dla (5.2) jest równy zeru. Macierz ta jest macierzą symetryczną. Często ONN są uczone za pomocą prostego algorytmu najszybszego spadku. Dzięki niewystępowaniu minimów lokalnych wartości oczekiwanych wielu funkcji celu, trening sieci tym algorytmem przebiega szybko. Wymaga on bardzo niewielkiej liczby obliczeń numerycznych. Stanowi to jego istotną zaletę, zwłaszcza jeżeli ma być wykorzystywany w pracy urządzeń, które nie dysponują

dużą mocą obliczeniową. Na rysunkach (5.3) i (5.4) przedstawiono porównanie szybkości uczenia FSNN algorytmem najszybszego spadku i sigmoidalnej sieci neuronowej, do której uczenia zastosowano algorytm Levenberga–Marquardta. Sieci uczono odwzorowywać następujące funkcje:

1) $g(x_1, x_2) = \exp\left(-(x_1 - 2)^2 - (x_2 - 1,5)^2\right)$
   $- \exp\left(-(x_1 - 4)^2 - (x_2 - 4)^2\right)$,
2) $g(x_1, x_2) = \mathrm{tgh}(0,8x_1 + x_2 - 6) - \mathrm{tgh}(1,25x_1 + 1,5x_2 - 9)$, gdzie tgh oznacza tangens hiperboliczny.

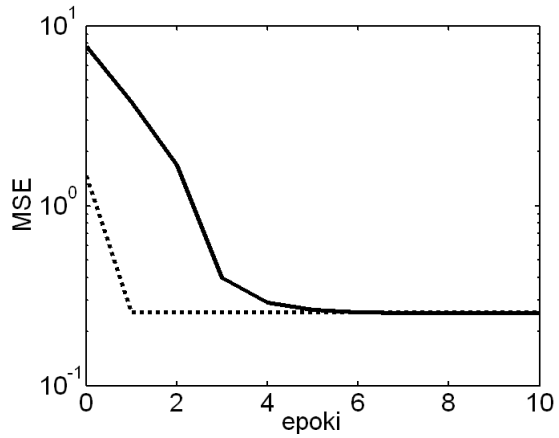


Rysunek 5.3. Porównanie szybkości uczenia funkcji nr 1 przez ONN trenowaną algorytmem najszybszego spadku (przerywana linia) i sieć sigmoidalną trenowaną za pomocą algorytmu Levenberga–Marquardta (ciągła linia)

W warstwie ukrytej sigmoidalnej sieci neuronowej znajdowało się 18 neuronów z funkcją aktywacji

$$f(u) = \frac{2}{1 + \exp(-2u)} - 1.$$

W warstwie wyjściowej obu sieci był neuron liniowy. FSNN miała 49 neuronów w warstwie ukrytej ($M_1 = M_2 = 7$). Liczbę neuronów dobrano tak, aby liczby wag w obu sieciach były podobne. Zbiór danych testowych składał się z par $\{y_k, [x_{1,k}, x_{2,k}]^T\}$, gdzie $y_k = \hat{y}(x_{1,k}, x_{2,k}) + \epsilon_k$, $x_{1,k}$ oraz $x_{2,k}$ były realizacjami zmiennej losowej o rozkładzie jednostajnym na przedziale $[0, 2\pi)$, ϵ_k były realizacjami zmiennej losowej o rozkładzie normalnym $\mathcal{N}(0,\ 0,25)$. Przed rozpoczęciem uczenia wszystkie wagi ONN były równe 1. Początkowe wartości wag sigmoidalnej sieci dobrano za pomocą algorytmu Nguyena–Widrowa. Wyniki uczenia były



Rysunek 5.4. Porównanie szybkości uczenia funkcji nr 2 przez ONN trenowaną algorytmem najszybszego spadku (przerywana linia) i sieć sigmoidalną trenowaną za pomocą algorytmu Levenberga–Marquardta (ciągła linia)

oceniane na zbiorze testowym, który został wygenerowany w sposób analogiczny do zbioru uczącego. Uczenie odwzorowywania podanych funkcji powtórzono 15 razy. Po każdej kolejnej epoce wartość MSE była estymowana na podstawie uśredniania 15 błędów średniokwadratowych.

Jeżeli podczas uczenia algorytmem najszybszego spadku pożądane jest zmniejszenie wpływu elementów odstających, wśród których mogą występować błędy grube [19], to można skorzystać z funkcji celu

$$E_2 = \sum_{k=1}^{N} \ln\left(1 + 0,5(y_k - \hat{y}(\mathbf{x}_k))^2\right). \tag{5.4}$$

Do redukcji maksymalnej wartości bezwzględnej błędu

$$\max_k |(y_k - \hat{y}(\mathbf{x}_k))|$$

często stosowana jest funkcja celu

$$E_3 = \frac{1}{2}\sum_{k=1}^{N}(y_k - \hat{y}(\mathbf{x}_k))^{2a}, \tag{5.5}$$

gdzie a jest liczbą naturalną większą od 1 [15].

W [8] wykazano, że dla FSNN w każdej iteracji algorytmu najszybszego spadku możliwe jest znaczne zredukowanie liczby wykonywanych operacji matematycznych przez obliczanie w odpowiedni sposób maksymalnej wartości współczynnika uczenia η, dla którego proces uczenia jest stabilny. Zaprezentowano

wzór umożliwiający łatwe jego wyznaczanie dla różnych funkcji celu. Pokazano, że dla (5.4) współczynnik uczenia jest ograniczony przez

$$\eta(t) < \frac{2 + r(t)^2}{\mathbf{v}(t)^T \mathbf{v}(t)}, \tag{5.6}$$

gdzie t jest numerem iteracji, $\mathbf{v}(t) = [\vartheta_1(t), \ldots, \vartheta_M(t)]^T$, ϑ_i jest wyjściem i-tego węzła mnożacego, $r(t)$ jest błędem będącym różnicą pomiędzy wartością zadaną ze zbioru uczącego a wartością wyjścia sieci. W przypadku funkcji celu (5.5), żeby proces uczenia był stabilny, musi być spełniony warunek

$$\eta(t) < \frac{2}{a\mathbf{v}(t)^T \mathbf{v}(t) r(t)^{2(a-1)}}, \tag{5.7}$$

Dla funkcji celu (5.2) warunek jest określony wzorem

$$\eta(t) < \frac{2}{\mathbf{v}(t)^T \mathbf{v}(t)}. \tag{5.8}$$

Ze względu na błędy numeryczne oraz aproksymowanie pewnych wielkości użytych w wyprowadzeniu (5.6), (5.7) i (5.8) współczynnik uczenia powinien być co najmniej o kilka procent mniejszy od prawej strony odpowiedniej nierówności. Zauważono, że dla FSNN wartość iloczynu $\mathbf{v}(t)^T \mathbf{v}(t)$, który występuje w mianowniku (5.6), (5.7) i (5.8), nie zależy od t.

Jeżeli wybrana jest funkcja celu (5.2), to spostrzeżenie umożliwia obliczenie współczynnika uczenia tylko raz, przed rozpoczęciem uczenia sieci. Uniknięcie ciągłego obliczania $\mathbf{v}(t)^T \mathbf{v}(t)$ umożliwia zmniejszenie liczby działań, również gdy funkcją celu jest (5.4) lub (5.5). W przypadku rozważanych funkcji do wyznaczenia zmian wag niezbędne jest w każdej iteracji wyznaczenie elementów wektora $\mathbf{v}$. Na komputerach, które nie mogą wykonywać równolegle wiele operacji zmiennoprzecinkowych, obliczanie $\mathbf{v}$ można przyspieszyć korzystając z zależności rekurencyjnych

$$\sin(\alpha x_i) = \sin((\alpha - 1)x_i)\cos(x_i) + \cos((\alpha - 1)x_i)\sin(x_i), \tag{5.9}$$

$$\cos(\alpha x_i) = \cos((\alpha - 1)x_i)\cos(x_i) - \sin((\alpha - 1)x_i)\sin(x_i), \tag{5.10}$$

gdzie α jest liczbą naturalną większą niż 1. Jeżeli $n = 1$ i funkcją celu jest (5.2), to przez uniknięcie konieczności ciągłego obliczania współczynnika uczenia można zredukować liczbę mnożeń w przybliżeniu o 20% i liczbę dodawań o około 25% [8].

Liczne modyfikacje algorytmu najszybszego spadku są używane w różnorodnych dziedzinach nauki. Do uczenia predyktorów neuronowych przewidujących

kilka kroków naprzód, które zostały zbudowane z predyktorów przewidujących w krótszej perspektywie czasu, powszechnie stosowany jest algorytm wstecznej propagacji w czasie BPTT (ang. *Backpropagation Through Time*) lub algorytm rekurencyjnego uczenia w czasie rzeczywistym RTRL (ang. *Real Time Recurrent Learning*), który cechuje się mniejszą złożonością obliczeniową od BPTT i wymaga mniejszej pojemności pamięci [14].

Jeżeli dysponujemy całym zbiorem uczącym przed rozpoczęciem uczenia, to w celu zminimalizowania (5.2) wagi ONN możemy obliczyć za pomocą metody najmniejszych kwadratów LSM (ang. *Least Squares Method*). Wówczas wektor wag jest równy

$$\mathbf{w} = (\theta^T\theta)^{-1}\theta^T\mathbf{y}, \tag{5.11}$$

gdzie

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \qquad \theta = \begin{bmatrix} \vartheta_1(\mathbf{x}_1) & \vartheta_2(\mathbf{x}_1) & \cdots & \vartheta_M(\mathbf{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ \vartheta_1(\mathbf{x}_N) & \vartheta_2(\mathbf{x}_N) & \cdots & \vartheta_M(\mathbf{x}_N) \end{bmatrix},$$

$\vartheta_i(\mathbf{x}_k)$ oznacza wartość wyjścia i-tego węzła mnożącego, gdy wejścia sieci są równe elementom wektora $\mathbf{x}_k$.

Jeżeli sieć pracuje w układzie adaptacyjnym, gdzie dane $\{\mathbf{x}_k, y_k\}$ są dostarczane w kolejnych chwilach, to dogodnie jest użyć rekurencyjnej metody najmniejszych kwadratów [21].

W celu poprawy zdolności generalizacji sieci oraz zmniejszenia wpływu błędów numerycznych podczas obliczania $\mathbf{w}$, zamiast LSM można zastosować estymatory obciążone, np. estymator grzbietowy (ang. *Ridge Estimator*) lub estymator Jamesa–Steina. Estymator grzbietowy w literaturze polskiej bywa także nazywany estymatorem grzebieniowym. Termin taki jest używany np. przez autora pracy [17]. Powstały liczne modyfikacje tego algorytmu [5].

Jedna z metod wykrywania elementów odstających polega na analizie wartości elementów wektora $\mathbf{r} = \mathbf{y} - \theta\mathbf{w}$. W celu wyeliminowania części elementów odstających można użyć następującego sposobu:

a) obliczyć $\mathbf{w}$ oraz $\mathbf{r}$,
b) powtórnie obliczyć $\mathbf{w}$, pomijając te pary $\{\mathbf{x}_k, y_k\}$, $k = 1, 2, \ldots, N$, które odpowiadają elementom $\mathbf{r}$ o dużych wartościach (sposób znajdowania tych elementów został opisany w [5]).

Postępując zgodnie z metodą przedstawioną w podanych punktach, pomija się elementy odstające, które niekoniecznie muszą być błędami grubymi. W ten sposób można nie uwzględnić sporej liczby interesujących danych. Dlatego zamiast ignorować całkowicie elementy odstające, niekiedy lepiej jest uwzględniać je w różnym stopniu. W tym celu można skorzystać z ważonej metody najmniej-

szych kwadratów WLSM (ang. *Weighted Least Squared Method*) [1]. Wektor wag obliczony za pomocą WLSM jest równy

$$\mathbf{w}_{WLSM} = (\theta^T \Omega \theta)^{-1} \theta^T \Omega \mathbf{y}, \tag{5.12}$$

gdzie $\Omega = \mathrm{diag}(\omega_1, \ldots, \omega_N)$, przy czym Ω jest macierzą diagonalną, jeżeli zakłócenia $\varepsilon_1, \ldots, \varepsilon_N$ nie są ze sobą skorelowane. Niech σ_r oznacza wartość odchylenia standardowego wyznaczoną za pomocą pewnego estymatora odpornego na elementy odstające.

W pracy [5] przedstawiono kilka takich estymatorów oraz podano informację, że elementy macierzy Ω mogą przyjmować wartości

$$\omega_i = \begin{cases} \frac{\psi(r_k/\sigma_r)}{r_k/\sigma_r} & \text{dla } r_k \neq 0, \\ 1 & \text{dla } r_k = 0, \end{cases} \tag{5.13}$$

gdzie r_k jest k-tym elementem wektora $\mathbf{r}$, $\psi(\cdot)$ jest funkcją, której wartość bezwzględna jest mniejsza bądź równa wartości bezwzględnej jej argumentu, i której wartość ma taki sam znak jak jej argument.

Do funkcji spełniających te warunki należą: funkcja Hampela, E_a funkcja Ramsey'a, sinusoidalna funkcja Andrewsa oraz funkcja Huberta [5] zdefiniowana przez

$$\psi(u) = \begin{cases} u & \text{dla } |u| \leqslant z, \\ z\,\mathrm{sign}(u) & \text{dla } |u| > z, \end{cases} \tag{5.14}$$

gdzie z jest pewną liczbą rzeczywistą. Wartość σ_r zależy od wektora $\mathbf{w}$. Iteracyjne obliczanie $\mathbf{w}$ i σ_r można wykorzystać do dalszego zmniejszenia wpływu błędów grubych. Niekiedy korzystne jest wykorzystanie zalet estymatora grzbietowego do obliczania wag sieci w sposób odporny na elementy odstające. W [17] zuważono, że istotą regresji grzbietowej jest „przeczesywanie" okolic oszacowania uzyskanego za pomocą LSM w następujący sposób:

$$\mathbf{w}_{RID} = (\theta^T \theta + cI_N)^{-1} \theta^T \mathbf{y}, \tag{5.15}$$

gdzie I_N oznacza macierz jednostkową stopnia N, c jest dodatnią liczbą rzeczywistą. Sposoby doboru wartości c zostały opisany w [10], [14]. Nietrudno jest spostrzec, że

$$\mathbf{w}_{RID} = (\theta^T \theta + cI_N)^{-1} \theta^T \theta \mathbf{w}, \tag{5.16}$$

gdzie $\mathbf{w}$ jest wektorem wag obliczonych za pomocą LSM. Zamieniając w (5.16) wektor $\mathbf{w}$ na $\mathbf{w}_{WLSM}$ otrzymujemy odporny estymator

$$\mathbf{w}_{RID2} = (\theta^T \theta + cI_N)^{-1} \theta^T \theta \mathbf{w}_{WLSM}, \tag{5.17}$$

który ma zalety estymatora grzbietowego. Prostym sposobem zmniejszenia wpływu błędów grubych, który często przynosi dobre rezultaty, jest uczenie sieci z wykorzystaniem funkcji celu będącej sumą wartości bezwzględnych błędów

$$E_4 = \frac{1}{2}\sum_{k=1}^{N} |y_k - \hat{y}(\mathbf{x}_k)|. \tag{5.18}$$

Do wad takiego podejścia należy brak możliwości wyprowadzenia wzoru, na podstawie którego można bezpośrednio określić wartości elementów wektora $\mathbf{w}$ minimalizującego (5.18) oraz brak gwarancji tylko jednego optymalnego rozwiązania. Jeżeli używana jest funkcji celu (5.18), to można wyznaczyć wektor wag za pomocą metod liniowego programowania.

Warto zwrócić uwagę, że istnieje wiele szczegółowo opisanych metod, które mogą być zastosowane do uczenia ONN, jeżeli dysponujemy pewną wiedzą *a priori* odnośnie do wartości wag. Ma to szczególne znaczenie, gdyż, jak już wcześniej zauważono, znacznie łatwiej jest dla ONN niż dla sieci sigmoidalnych zinterpretować znaczenie wartości poszczególnych wag sieci neuronowych, które modelują pewne zjawiska fizyczne.

W [9] zauważono, że gdy $\mathbf{x}_k$ są odpowiednio rozmieszczone w n-wymiarowym hipersześcianie $\mathcal{H}$, którego długość boków jest równa 2π, to można obliczyć wagi FSNN, stosując bezpośrednio metodę, która wymaga znacznie mniejszych nakładów obliczeniowych niż LSM. W [7] pokazano podobną procedurę, która umożliwia zmniejszenie wpływu błędów grubych. Wykorzystuje ona efektywne algorytmy wyznaczające dyskretną transformatę Fouriera DFT (ang. *Discrete Fourier Transform*) za pomocą szybkiej transformaty Fouriera w sposób przedstawiony dalej. Opis algorytmów FFT można znaleźć w [2], [22], [24], [25].

Jeżeli wektory $\mathbf{x}_k$ są równomiernie rozmieszczone w hiperprostopadłościanie lub hipersześcianie, mającym inną długość boków, to wystarczy przeskalować $\mathbf{x}_k$, przemnażając poszczególne ich elementy przez odpowiednie stałe.

Niech $U_m \in \mathbb{R}^n$, $m = 1, 2, \ldots$ będą wszystkimi wektorami, których i-ty element może przyjmować wartości z szeregu $\{0, p_i, 2p_i, \ldots, M_i - 1\}$, gdzie $p_i = 2\pi/M_i$. Takich wektorów jest M. Elementy tych wektorów są równe współrzędnym punktów rozmieszczonych równomiernie w hipersześcianie $\mathcal{H}$ (punkty te leżą na n wymiarowej hiperkracie). Załóżmy, że w zbiorze uczącym dysponujemy dla każdego U_m co najmniej b_m wartościami $y_{m,a} = \hat{y}(U_m) + \varepsilon_{m,a}$, gdzie $b_m \geqslant 1$, $a = 1, 2, \ldots, b_m$, $\varepsilon_{m,a}$ są zmiennymi losowymi o zerowej wartości oczekiwanej i skończonej wariancji. Oznaczmy

$$\overline{f}(U_m) = \frac{y_{m,1} + y_{m,2} + \ldots + y_{m,b_m}}{b_m}.$$

Jeżeli wśród danych $y_{m,1}, \ldots, y_{m,b_m}$ mogą pojawić się błędy grube, to zaleca się, żeby $\overline{f}(U_m)$ było równe medianie lub średniej windsorskiej z $y_{m,1}, y_{m,2}, \ldots, y_{m,M_m}$.

n-wymiarowa DFT z $\overline{f}(U_1), \overline{f}(U_2), \ldots$ jest określona wzorem

$$\begin{aligned} F(k_1, \ldots, k_n) &= \sum_{u_1=0}^{M_1-1} \sum_{u_2=0}^{M_2-1} \ldots \sum_{u_n=0}^{M_n-1} \overline{f}\left([u_1, u_2, \ldots, u_n]^T\right) \\ &\quad e^{-j(k_1u_1p_1+\ldots+k_nu_np_n)}, \end{aligned} \tag{5.19}$$

gdzie $k_1, k_2, \ldots, k_n$ są nieujemnymi liczbami całkowitymi, mniejszymi odpowiednio od $M_1, M_2, \ldots, M_n$, $j = \sqrt{-1}$.

Stosując odwrotną dyskretną transformatę Fouriera IDFT (ang. *Inverse Discrete Fourier Transform*), można napisać [7]

$$\begin{aligned} f\left(\frac{u_1}{p_1}, \ldots, \frac{u_n}{p_n}\right) &\approx \frac{1}{M} \sum_{k_1=0}^{M_1-1} \sum_{k_2=0}^{M_2-1} \ldots \sum_{k_n=0}^{M_n-1} F\left(k_1, \ldots, k_n\right) \\ &\quad e^{j(k_1u_1p_1+\ldots+k_nu_np_n)} \end{aligned} \tag{5.20}$$

Dokładność aproksymacji (5.20) zależy od parametrów addytywnego szumu ε_i i liczb $M_1, \ldots, M_n$ oraz $b_1, \ldots, b_M$. Współczynniki DFT obliczonej z danych rzeczywistych mają właściwość symetrii

$$\begin{aligned} F(M_1 - k_1, M_2 - k_2, \ldots, M_n - k_n) &= F^*(k_1, k_2, \ldots, k_n), \\ F(k_1, M_2 - k_2, \ldots, M_n - k_n) &= F^*(M_1 - k_1, k_2, \ldots, k_n), \\ &\vdots, \end{aligned} \tag{5.21}$$

gdzie * oznacza liczbę sprzężoną. Wykorzystując wzory Eulera oraz (5.21), można (5.20) przekształcić do postaci

$$\begin{aligned} &f(u_1p_1, \ldots, u_np_n) \approx \\ &\quad \approx s + \frac{2}{M} \sum_{k_1=1}^{M_1/2-1} \sum_{k_2=1}^{M_2/2-1} \ldots \sum_{k_n=1}^{M_n/2-1} Re\left\{F(k_1, k_2, \ldots, k_n)\right\} \\ &\quad \cos\left(p_1k_1u_1 + p_2k_2u_2 + \ldots + p_nk_nu_n\right)) \\ &\quad -Im\left\{Y(k_1, k_2, \ldots, k_n)\right\} \\ &\quad \sin\left(p_1k_1u_1 + p_2k_2u_2 + \ldots + p_nk_nu_n\right) \\ &\quad +Re\left\{Y(k_1, k_2, \ldots, M_n - k_n)\right\} \\ &\quad \cos\left(p_1k_1u_1 + p_2k_2u_2 + \ldots + p_{n-1}k_{n-1}u_{n-1} - p_nk_nu_n\right) \end{aligned} \tag{5.22}$$

$$
\begin{aligned}
&-Im\left\{Y(k_1,k_2,\ldots,M_n-k_n)\right\}\\
&\sin\left(p_1k_1u_1+p_2k_2u_2+\ldots+p_{n-1}k_{n-1}u_{n-1}-p_nk_nu_n\right)\\
&+\ldots+\\
&+Re\left\{Y(k_1,M_2-k_2,\ldots,M_n-k_n)\right\}\\
&\cos\left(p_1k_1u_1-p_2k_2u_2-\ldots-p_nk_nu_n\right)\\
&-Im\left\{Y(k_1,M_2-k_2,\ldots,M_n-k_n)\right\}\\
&\sin\left(p_1k_1u_1-p_2k_2u_2-\ldots-p_nk_nu_n\right),
\end{aligned}
$$

gdzie s jest sumą kombinacji wszystkich sum

$$
\sum_{k_1=1}^{M_1/2-1}\sum_{k_2=1}^{M_2/2-1}\ldots\sum_{k_n=1}^{M_n/2-1} Re\left\{F(k_1,k_2,\ldots,k_n)\right\}e^{j(p_1k_1u_1+\ldots+p_2k_2u_2)},
$$

w których co najmniej jeden element wektora $[k_1,\ldots,k_n]^T$ jest równy zeru, i w których pominięto symbole sumy odpowiadające tym zerowym elementom.

Na podstawie wzoru (5.22) można w efektywny sposób wyznaczyć wagi sieci. Jeśli najpierw zamieni się we wzorze (5.22) $u_1,\ldots,u_n$ na $x_1,\ldots,x_n$, a następnie rozpisze funkcje sinus i cosinus, których argumenty są sumą ważoną kilku wejść sieci, na sumę iloczynów funkcji sinus i cosinus, których argumenty zależą jedynie od pojedynczych wejść, to nietrudno wtedy spostrzec, że iloczyny te mają takie same wartości jak wyjścia węzłów mnożących FSNN, gdy $c_1=1$ i $c_2=1$. Aby zatem wyznaczyć wartości poszczególnych wag, wystarczy zsumować wszystkie współczynniki DFT znajdujące się przed iloczynami identycznych funkcji, które mają takie same argumenty. W [9] spostrzeżono, że w łatwy sposób można utworzyć program komputerowy, który podawałby symboliczne wyrażenia na wagi.

Podany sposób ustalania wartości wag cechuje się mniejszą złożonością obliczeniową niż wyznaczanie wektora w za pomocą LSM. Jeżeli $\mathbf{x}_k$ nie są równomiernie rozmieszczone w hiperprzestrzeni, to jedno z możliwych rozwiązań do zastosowania w takiej sytuacji polega na użyciu nierównomiernej szybkiej transformaty Fouriera NUFFT (ang. *Nonuniform Fast Fourier Transform*) [13] zamiast FFT do obliczania rozważanych współczynników. Inne sposoby postępowania mogą opierać się na aproksymowaniu $\overline{f}(U_1),\overline{f}(U_2),\ldots,$ na podstawie wartości $\mathbf{y}_1,\ldots,\mathbf{y}_N$. W [9] szczegółowo przedstawiono szybki sposób aproksymacji $\overline{f}(U_1),\overline{f}(U_2),\ldots$ z użyciem średnich arytmetycznych z $\mathbf{y}_1,\ldots,\mathbf{y}_N$ należących do małych rozłącznych hipersześcianów. Zaprezentowano wyniki symulacji komputerowej, w której użyto takiej aproksymacji. W [7] zauważono, że rozważaną aproksymację można również wykonać za pomocą odpowiednich, nieskomplikowanych hiperpowierzchni. Nierzadko w praktyce inżynierskiej spotyka się dane niezależne, które są rozmieszczone równomiernie, zwłaszcza że zgodnie z teorią

planowania eksperymentu [17] sytuacja z takim rozłożeniem nośników planu ma kilka ważnych zalet.

Gdyby liczba elementów wektora wag była taka sama jak liczba danych, to FSNN nie miałaby zdolności generalizacji. Należałoby wówczas wykonać regularyzację sieci, np. przez pominięcie wag o małych wartościach. W [9] pokazano przykład regularyzacji za pomocą takiej metody.

5.4. Szybkie obliczanie wartości wyjść FSNN

W [6] przedstawiono efektywny sposób obliczania wartości wyjścia MISO FSNN równocześnie dla wielu różnych wartości wejść wraz z propozycjami jego zastosowania. Jedna z nich dotyczy wykonywania regularyzacji przez wczesne zatrzymanie [14].

n-wymiarowa DFT z przekształcenia danych wejściowych, które wykonuje sieć, jest równa

$$\hat{Y}(k_1,\ldots,k_n) = \sum_{u_1=0}^{M_1-1}\sum_{u_2=0}^{M_2-1}\cdots\sum_{u_n=0}^{M_n-1}\hat{y}\left([x_1,x_2,\ldots,x_n]^T\right) e^{-j(k_1x_1x_1+\ldots+k_nx_np_n)} \tag{5.23}$$

Jeżeli znamy wartości współczynników $\hat{Y}$, to w celu otrzymania wartości wyjścia sieci jednocześnie dla M wektorów wejściowych, których elementy są równe współrzędnym punktów leżących na hiperkracie opisanej w podrozdziale 5.3, można obliczyć IDFT

$$\hat{y}\left(\frac{u_1}{p_1},\ldots,\frac{u_n}{p_n}\right) \approx \frac{1}{M}\sum_{k_1=0}^{M_1-1}\sum_{k_2=0}^{M_2-1}\cdots\sum_{k_n=0}^{M_n-1}\hat{Y}\left(k_1,\ldots,k_n\right) e^{j(k_1u_1p_1+\ldots+k_nu_np_n.)} \tag{5.24}$$

Zaproponowana w [6] metoda wyznaczania wartości wyjścia MISO FSNN dla różnych wartości wejść wymaga wykonania dwóch kroków:

- obliczenie współczynników określonych równaniem (5.23) na podstawie wartości wag sieci,
- wyznaczenie odwrotnej DFT za pomocą algorytmów IFFT.

Obliczenie jednego współczynnika DFT wymaga wykonania $O(2^n)$ działań. Wystarczy obliczyć tylko współczynniki DFT dla $k_1 < M_1/2+1,\ldots,k_n < M_n/2+1$, ponieważ pozostałe $\hat{Y}(k_1,\ldots,k_n)$ są liczbami sprzężonymi. Złożoność obliczeniowa n wymiarowej IFFT wynosi

$$O\left(\sum_{v=1}^{n}\left(\prod_{i=1}^{n}M_i\right)\log_2 M_v\right).$$

Całkowita zatem złożoność proponowanej metody to

$$O\left(2^n\left(\prod_{v=1}^{n} M_v\right)+\sum_{v=1}^{n}\left(\prod_{i=1}^{n} M_i\right)\log_2 M_v\right).$$

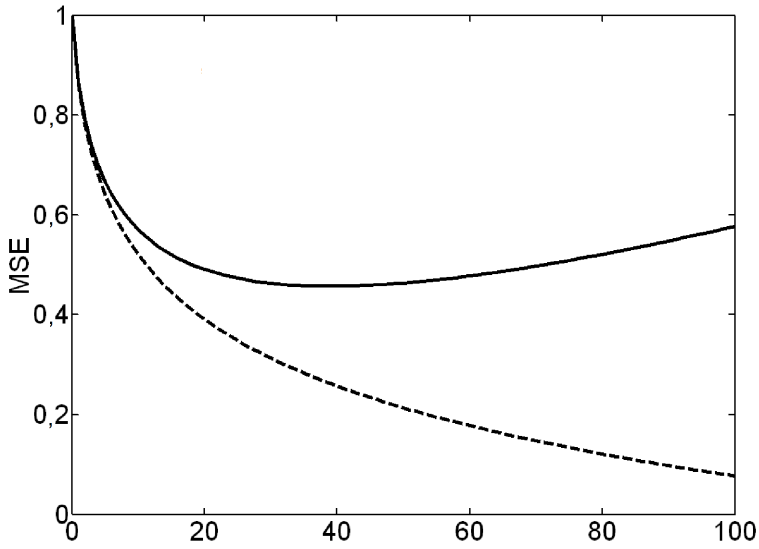
Aby dokonać tych samych obliczeń bezpośrednio na podstawie równania (5.1), należałoby wykonać w przybliżeniu aż $O(M^2)$ działań.

W [6] zauważano, że w celu wyznaczenia wartości wyjścia sieci dla wektorów wejściowych, których elementy są równe współrzędnym punktów leżących na przesuniętej hiperkracie, należy współczynniki DFT przemnożyć przez liczbę Eulera podniesioną do odpowiedniej potęgi. Załóżmy, że przesunięcie wspomnianej hiperkraty względem punktu, którego wszystkie współrzędne są równe zeru, wynosi w i-tym wymiarze δ_i, gdzie $\delta_1, \ldots, \delta_n$ są liczbami rzeczywistymi. W takiej sytuacji należy współczynniki $\hat{Y}(k_1, \ldots, k_n)$ przemnożyć przez

$$\prod_{i=0}^{n} e^{jp_i k_i \delta_i}.$$

Wykonując obliczenia dla kilku przesuniętych hiperkrat, można zmniejszyć odległości pomiędzy rozważanymi punktami.

Metoda opisana w [6] może być wykorzystana w trakcie uczenia do śledzenia zmian przekształcenia danych wejściowych wykonywanego przez sieć. Zwłaszcza w układach adaptacyjnych, gdzie FSNN jest bez przerwy douczana, monitorowanie działania sieci często jest niezbędne. Przedstawiona metoda nadaje się do tworzenia dwuwymiarowych i trójwymiarowych wykresów $\hat{y}$. Można ją także wykorzystać do wykonania regularyzacji przez wczesne zatrzymanie, jeżeli zbiór $\{\mathbf{x}_k, y_k\}$ da się podzielić na zbiór testowy i zbiór danych do uczenia sieci w taki sposób, żeby elementy wektorów $\mathbf{x}_k$ ze zbioru testowego były równe współrzędnym punktów leżących na n wymiarowej hiperkracie. Wówczas po każdej epoce uczenia (prezentacji wszystkich wektorów $\mathbf{x}_k$ ze zbioru do uczenia) należy w opisany sposób wyznaczyć wartości wyjścia sieci dla wszystkich $\mathbf{x}_k$ ze zbioru testowego. Wyniki te są niezbędne do obliczenia wybranej funkcji celu. Gdy wartość tej funkcji wzrasta, proces uczenia jest zatrzymywany. Typowy kształt zmian MSE odwzorowania wykonywanego przez sieć dla danych testowych i danych użytych do uczenia [14] przedstawiono na rysunku 5.5. Jeżeli nie jest możliwy podział zbioru $\{\mathbf{x}_k, y_k\}$ we wspomniany sposób, to można albo wykorzystać NUFFT, albo aproksymować wartości y_k odpowiadające punktom leżącym na wspomnianej hiperkracie i następnie z tych przybliżeń obliczyć IFFT. Szczególnie interesujące wydaje się połączenie przedstawionej metody z algorytmem najszybszego spadku. Mała liczba obliczeń jest wtedy potrzebna zarówno do wyznaczania zmian wag w kolejnych iteracjach, jak i do określenia właściwego momentu zakończenia treningu FSNN.



Rysunek 5.5. Zależność MSE od długości procesu uczenia dla zbioru danych uczących (przerywana linia) i zbioru testowego (ciągła linia)

5.5. Podsumowanie

ONN doskonale nadają się do wykorzystania w wielu aplikacjach, gdzie stosowane są sieci sigmoidalne, które mają niewielka liczbę wejść. Istotną wadą ONN jest wykładniczy wzrost liczby neuronów wraz ze zwiększeniem się liczby wejść sieci. Dlatego zaleca się przeanalizowanie możliwości zastosowania wraz z ONN metod redukcji wymiaru danych wejściowych, które zostały przedstawione między innymi w [4], [11], [16]. Bardzo interesujący sposób zmniejszenia wymiaru danych za pomocą głównych krzywych został pokazany w [12].

Dzięku dużej szybkości uczenia i małej liczbie potrzebnych obliczeń arytmetycznych FSNN wraz z algorytmem najszybszego spadku doskonale nadają się do pracy w układach elektronicznych, w których dostępna moc obliczeniowa jest zbyt mała do uczenia innych rodzajów sieci neuronowych. Stosunkowo duża liczba różnych algorytmów, które mogą być wykorzystane do redukcji wpływu elementów odstających, umożliwia wykorzystanie ONN do przetwarzania danych, wśród których mogą pojawić się błędy grube.

Nietrudno jest napisać program przeznaczony do szybkiego wyznaczania wag lub obliczania wartości wyjścia FSNN za pomocą FFT lub IFFT. Istnieje wiele

gotowych bibliotek z procedurami liczącymi FFT. Prawdopodobnie najszybszą, niezależną od sprzętu, biblioteką dostępną na licencji GNU General Public License jest FFTW (ang. *Fastest Fourier Transform in the West*), którą można pobrać ze strony http://www.fftw.org. Dostępne są także wersje dla komputerów wieloprocesorowych oraz do obliczeń rozproszonych. Wielowymiarową FFT można wyznaczyć przez wielokrotne obliczanie jednowymiarowych DFT. Do otrzymania współczynników DFT lub IDFT z danych rzeczywistych zaleca się zastosować algorytm „2N punktowej rzeczywistej FFT". Umożliwia on zmniejszenie liczby mnożeń o 30 procent.

Sprzętowa realizacja sieci neuronowych umożliwia w pełni wykorzystać potencjał, jaki ma ich równoległa architektura. Na rynku dostępnych jest wiele układów scalonych, zawierających w sobie różne struktury sieci neuronowych. Brak jest jednak rozwiązań dedykowanych do ONN. Układy programowalne FPGA doskonale nadają się do wykorzystania sprzętowej implementacji ONN. Obecnie produkowane FPGA zawierają setki bloków mnożących, które potrafią wykonać miliardy obliczeń na sekundę. Układy te można wygodnie zaprogramować za pomocą języków opisu sprzętu, np. VHDL lub Verilog.

Bibliografia

[1] Björck A., *Numerical Methods for Least Squares Problems*, SIAM, Amsterdam, 1996.
[2] Chu E., George A., *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*, CRC Press, Boca Raton, 2000.
[3] Daniłow W., Iwanowa A., Isakowa K., Lusternik L., Salechow G., Chowański N., Cław J., Janpolski A., *Funkcje, granice, szeregi, ułamki łancuchowe*, PWN, Warszawa, 1970.
[4] Diamantaras K., Kung S., *Principal component neural networks – Theory and applications*, Springer, New York, 1996.
[5] Groß J., *Linear Regression*, Springer-Verlag, Berlin, 2003.
[6] Halawa K., *Szybka metoda obliczania wyjść fourierowskich sieci neuronowych* w *Sterowanie i automatyzacja: aktualne problemy i ich rozwiązania*, (red.) Krzysztofa Malinowskiego, Leszka Rutkowskiego, Exit, Warszawa, 2008, 652–659.
[7] Halawa K., *Fast and Robust Way of Learning the Fourier Series Neural Networks on the Basis of Multidimensional Discrete Fourier Transform* Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence, Proc. of 9th Int. Conf. on Artificial Intelligence and Soft Computing, 2008, Vol. 5097, 62—70.
[8] Halawa K., *Optymalizacja procesu uczenia fourierowskich sieci neuronowych przy wykorzystaniu algorytmu najszybszego spadku*, Przegląd Elektrotechniczny, 6/2008, 125–127.
[9] Halawa K., *Determining the Wegihts of a Fourier Series Neural Network on the Basis of Multidimensional Discrete Fourier Transform*, International Journal of Applied Mathematics and Computer Science, 2008, Vol. 18, No. 3, 369–375.

[10] Hansen Ch., *Rank-Deficient and Discrete Ill-Posed Problems – Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1998.
[11] Hyvärinen A. and Oja E., *Independent Component Analysis: Algorithms and Applications*, Neural Networks, 2000, Vol. 13, No. 4, 411–430.
[12] Kegl B., Krzyżak A., Linder T., Zeger K., *Learning and Design of Principal Curves*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, Vol. 22, No. 3, 281–297.
[13] Liu Q.H., Nyguen N., *An accurate algorithm for nonuniform fast Fourier transforms*, IEEE Microwave Guided Wave Letters, 1997, Vol. 8, No. 1, 18–20.
[14] Nelles 0., *Nonlinear System Identification From Classical Approaches to Neural Network and Fuzzy Model*, Springer-Verlag, Berlin, 2001.
[15] Osowski S., *Sieci neuronowe w ujęciu algorytmicznym*, WNT, Warszawa, 1997.
[16] Osowski S., *Sieci neuronowe do przetwarzania informacji*, Oficyna Wydawnicza Politechniki Warszawskiej, 2006.
[17] Rafajłowicz E., *Algorytmy planowania eksperymentu z implementacjami w środowisku Mathematica*, PLJ, Warszawa, 1996.
[18] Rafajłowicz E., Pawlak M., *On Function Recovery by Neural Networks Based on Orthogonal Expansions*, Nonlinear Analysis, Theory and Applications, 1997, Vol. 30, No. 3, 1343–1354, Proc. 2nd World Congress of Nonlinear Analysis, Pergamon Press.
[19] Rusiecki A., *Algorytmy uczenia sieci neuronowych odporne na błędy w danych*, Praca doktorska, Politechnika Wrocławska, 2007.
[20] Sher C.F, Tseng C.S. and Chen C.S., *Properties and Performance of Orthogonal Neural Network in Function Approximation* International Journal of Intelligent Systems, 2001, Vol. 16, No. 12, 1377–1392.
[21] Söderström T., Stoica P., *Identyfikacja systemów*, PWN, Warszawa, 1997.
[22] Stasiński R., *O liczeniu dyskretnej transformacji Fouriera*, Wydawnictwo Politechniki Poznańskiej, 2005.
[23] Tseng C.S., Chen C.S., *Performance Comparison between the Training Method and the Numerical Method of the Orthogonal Neural Network in Function Approximation*, International Journal of Intelligent Systems, 2004, Vol. 19, No. 12, 1257–1275.
[24] Van Loan Ch., *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.
[25] Walker J.S., *Fast Fourier Transforms, CRC Press*, Boca Raton, 1996.
[26] Zhu C., Shukla D., Paul F.W., *Orthogonal Functions for System Identification and Control*, Control and Dynamic Systems: Neural Network Systems Techniques and Applications, 2002, Vol. 7, 1–73 pod redakcją: Leondes C.T., Academic Press, San Diego.

Rozdział 6

Sieci kontekstowe i strumienie danych

Piotr Ciskowski

6.1. Wstęp

Wiele badań dowodzi, że ludzie zdobywają wiedzę, zapamiętują i przetwarzają informacje, a także rozumują w sposób kontekstowy. Pojęcia oraz wzorce zachowań mogą być przechowywane w ludzkim mózgu w postaci ustrukturyzowanej jako sekwencje zdarzeń występujących w konkretnych kontekstach. Rozwiązywanie skomplikowanego problemu przez człowieka często polega na zidentyfikowaniu kontekstu, w jakim on wystąpił, a następnie na rozwiązaniu znacznie prostszego zadania charakterystycznego dla danego kontekstu. Maszyny uczące często budowane są przez naśladowanie zjawisk występujących w naturze, a także na podstawie obserwacji procesów uczenia u zwierząt i ludzi. Identyfikacja kontekstu, dekompozycja zadań na mniejsze, a także integracja prostych rozwiązań w rozwiązania całościowe, pomaga efektywnie działać również skomplikowanym systemom uczącym przetwarzającym wiedzę o złożonych zjawiskach.

W niniejszej pracy przedstawione zostaną niektóre znane z literatury sposoby wykorzystania informacji kontekstowej w uczeniu maszynowym. Na ich tle opisane zostaną badane i rozwijane przez autora kontekstowe sieci neuronowe – model kontekstowego neuronu oraz struktury sieci kontekstowych, ich własności oraz algorytmy uczenia. Szczególnie dokładnie omówiony zostanie proces odwzorowania przestrzeni sygnałów wejściowych zawierającej zmienne kontekstowe w zmienne wyjściowe. Przeanalizowana zostanie przydatność sieci kontekstowych w przetwarzaniu intensywnych strumieni danych. Przedstawione zostaną przykłady działania sieci tradycyjnych oraz kontekstowych w zadaniach klasyfikacji oraz modelowania zależności finansowych.

6.2. Znaczenie kontekstu w modelowaniu

W wielu praktycznych zastosowaniach maszyn uczących spotykamy się z koniecznością modelowania zjawisk i zależności o złożonym charakterze. Zjawiska te, lub np. rozpoznawane obiekty, opisywane są za pomocą pewnego zestawu

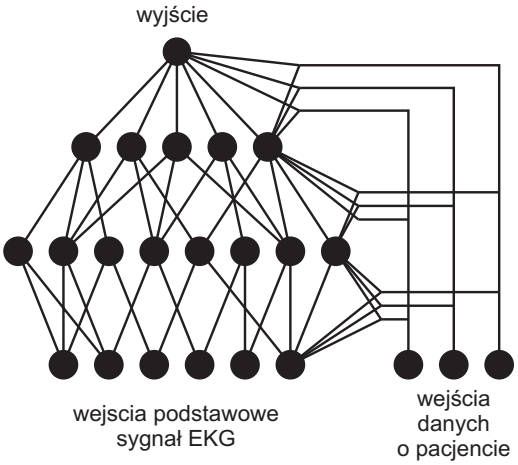
zmiennych, zwanych cechami. Działanie maszyny uczącej możemy rozumieć jako odwzorowanie zbioru cech obiektu w pewną zmienną wyjściową. Zakładamy, że zmienna wyjściowa może być zarówno dyskretna, jak i ciągła. W zależności od jej charakteru, rodzaj zadania wykonywanego przez maszynę uczącą określamy jako decyzyjny (klasyfikacja) lub funkcyjny (regresja, predykcja).

Często mamy do czynienia z sytuacją, gdy modelowane przez nas odwzorowanie podstawowych cech obiektu w zmienną wyjściową zmienia się w zależności od pewnych czynników zewnętrznych, czyli od zmiennych innych niż te bezpośrednio przetwarzane przez maszynę uczącą. Mogą one określać np. bieżący stan rozpoznawanego obiektu czy środowiska, w którym on się znajduje. Właśnie takie wielkości – zmienne *towarzyszące* – będziemy nazywali kontekstem.

Mówiąc bardziej formalnie, przez kontekst rozumiemy cechy badanego obiektu, nie przetwarzane bezpośrednio przez maszynę uczącą, lecz mające wpływ na postać realizowanego przez nią odwzorowania. Z dotychczasowych rozważań wynika już, że zmienne podstawowe są niezbędne do rozwiązania danego problemu – działanie maszyny uczącej przetwarzającej tylko cechy podstawowe będzie zadowalające. Natomiast zmienne kontekstowe mają jedynie charakter pomocniczy – nie można prawidłowo wykonać zadania wykorzystując tylko zmienne kontekstowe, można natomiast z ich pomocą polepszyć działanie maszyny uczącej przetwarzającej zmienne podstawowe. Znaczenie cech kontekstowych jest przy tym tak istotne, że zmieniają one sposób przetwarzania cech podstawowych.

Argumentem przemawiającym za koniecznością podziału zmiennych na podstawowe i kontekstowe jest również fakt, że zmienne kontekstowe często mają zupełnie inne znaczenie niż zmienne podstawowe. Na przykład, w pewnych zastosowaniach zmiennymi podstawowymi mogą być cechy obiektów rozpoznanych na obrazie z kamery przemysłowej, natomiast zmienną kontekstową – jasność obrazu oznaczająca porę dnia bądź zmienna ciągła, której wartość bezpośrednio oznacza porę dnia. W innym przypadku zmiennymi podstawowymi będą kolejne próbki sygnału EKG pacjenta, zmienną kontekstową zaś – jego wiek. Również z powodu tej odmienności znaczeń nie powinniśmy tak po prostu dodawać zmiennych kontekstowych do zmiennych podstawowych. Należy raczej modelowaną za pomocą maszyny uczącej zależność (żądanego wyjścia maszyny uczącej od podanych na wejście cech podstawowych) uczynić *funkcją* cech kontekstowych.

Intuicyjnie nasuwającą się dziedziną zastosowań dla modelowania kontekstowego jest wspomaganie podejmowania decyzji medycznych. Nie mamy tu przy tym na myśli systemów decyzyjnych wyższego poziomu, często złożonych z podsystemów analizujących różnego rodzaju informacje o stanie pacjenta, podejmujących złożone decyzje dotyczące stanu zdrowia czy sugerujących zalecaną terapię. Modelowanie kontekstowe, o którym mowa w niniejszym rozdziale, zachodzi na



Rysunek 6.1. Model sieci użytej w [13]

niższym poziomie – przetwarzania konkretnych sygnałów pomiarowych czy klasyfikacji obrazów i wzorców. Może więc znaleźć zastosowanie w specjalizowanych częściach wspomnianych większych systemów decyzyjnych.

Za przykład maszyny uczącej działającej w sposób kontekstowy niech posłuży sieć neuronowa, opisana w [13], analizująca przebieg czasowy sygnału EKG, czyli ciąg próbek sygnału w czasie (poddany pewnej wstępnej obróbce) i na jego podstawie sugerująca lekarzowi decyzję o normalnym lub chorobowym charakterze tego sygnału. Jej schemat przedstawiono na rysunku 6.1. Danymi podstawowymi są w tym przypadku kolejne próbki sygnału EKG. Lekarz, którego sposób postępowania próbuje naśladować taka sieć, jest w stanie podjąć decyzję o stanie zdrowia pacjenta tylko na podstawie sygnału EKG, nie mając żadnych dodatkowych informacji. Naturalne jest jednak, że wzorce sygnałów normalnych oraz chorobowych różnią się w zależności od pewnych czynników, np. od wieku pacjenta. Lekarz podświadomie używa tej informacji podczas podejmowania decyzji – w zależności od niej zwraca uwagę na różne szczegóły sygnału EKG. Może na przykład pewne zmiany uznać za dopuszczalne u starszego pacjenta, niepokojące zaś u młodszego. Wiek nie jest zmienną podstawową, gdyż nie można podjąć decyzji o stanie pacjenta jedynie na podstawie jego wieku. Jest to zmienna pomocnicza, której użycie udoskonala decyzję podjętą tylko na podstawie analizy sygnału EKG. Widać przy tym, że z racji odrębnego charakteru wartości zmiennej *wiek* nie powinny być przetwarzane w taki sam sposób, jak wartości kolejnych próbek sygnału EKG. Powinny natomiast w bardziej ogólnym sensie wpływać na sposób przetwarzania próbek sygnału EKG przez sieć.

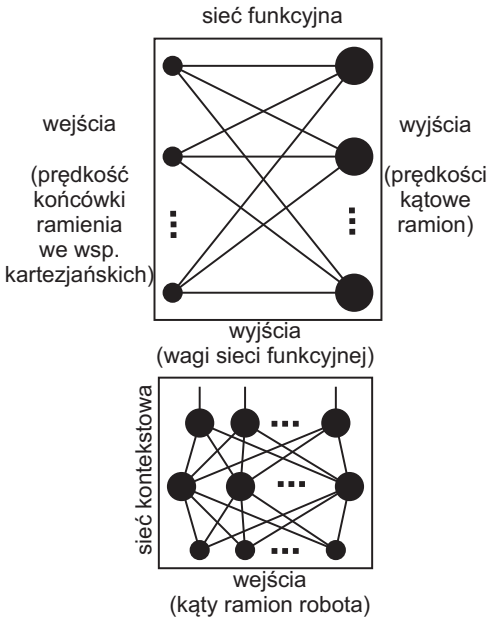
Jedną z pierwszych prac, w których zwrócono uwagę na możliwość kontekstowego podejścia do tego problemu, była [13]. Użyto w niej sieci neuronowych do klasyfikacji sygnałów EKG w sposób opisany wcześniej. Jak zauważają autorzy, jednym z czynników utrudniających ciągłe monitorowanie tych sygnałów jest nie tylko ich zmienność morfologiczna pomiędzy pacjentami, lecz także zmienność sygnałów w czasie u tego samego pacjenta. Zaproponowany system składał się z neuronowego klasyfikatora sygnałów EKG oraz modułu pacjenta. Sieć neuronowa działała tylko na próbkach sygnału EKG, nie biorąc pod uwagę kontekstowych danych o pacjencie, natomiast moduł pacjenta służył jako dodatkowe wejścia mnożące, dostosowujące decyzję sieci do bieżącego stanu konkretnego pacjenta.

W niniejszym rozdziale przestawiono model sieci neuronowej o podobnych założeniach, lecz inaczej modelowanej zależności wag prowadzących do wejść podstawowych sieci od kontekstu. Wagi te będą funkcyjnie zależne od wartości wejść kontekstowych. Przeanalizowano również proces kontekstowej klasyfikacji w sieciach tradycyjnych oraz kontekstowych.

Inną dziedziną zastosowań modelowania kontekstowego jest identyfikacja systemów oraz sterowanie. W tym przypadku motywacja do użycia kontekstowych sieci neuronowych wynika często nie z interpretacji znaczenia poszczególnych wejść systemu, lecz z charakteru modelowanych zależności matematycznych między wejściami a wyjściami systemu. Takie podejście do problemu znajdziemy w przykładach sterowania ramieniem robota oraz modelowania tłumika drgań sejsmicznych, przedstawionych w [12] i [14].

Pierwszy z nich był bezpośrednim bodźcem do opracowania modeli kontekstowych sieci neuronowych przedstawionych w niniejszej pracy. Kontekstowy sposób rozwiązywania złożonych problemów sterowania wynika z wykorzystania strategii dziel-i-rządź (ang. *divide-and-conquer*), polegającej na podziale rozwiązywanego problemu na mniejsze podproblemy. W tym przypadku skomplikowana zależność wejście–wyjście modelowanego systemu dzielona jest na prostsze zależności występujące w konkretnych kontekstach, w jakich może się znaleźć system. Pomimo że system uczący musi dodatkowo pamiętać przyporządkowanie podproblemów do konkretnych kontekstów, podejście to pozwala uprościć i przyspieszyć rozwiązywanie skomplikowanych zadań. W pracy [14] po raz pierwszy użyto pojęcia *context-sensitive networks*, sieci neuronowych czułych na kontekst.

Wspomnianą metodę *dziel i rządź* zastosowano tu do wyznaczania kinematyki odwrotnej, czyli przekształcania prędkości końcówki ramienia robota we współrzędnych kartezjańskich na prędkości kątowe jego ramion. Zależność prędkości kątowych od prędkości liniowych jest ściśle zależna od bieżącego położenia ramienia robota. Ta dwustopniowa zależność znalazła dokładne odzwierciedlenie w użytym modelu sieci neuronowych. Składał się on z dwóch sieci neuronowych: funkcyjnej



Rysunek 6.2. Model sieci użytej w [14]

oraz kontekstowej. Pierwsza z nich uczy się zależności prędkości kątowych od prędkości liniowych w danym kontekście, jakim jest aktualna pozycja ramienia robota. Druga sieć ustawia wagi pierwszej sieci w zależności od kontekstu.

W tym rozdziale pokażemy, że zaproponowany przez nas model kontekstowych sieci neuronowych pozwoli zintegrować modelowanie podobnych dwuetapowych zależności w jednej sieci: zależność wyjść sieci od wejść – za pomocą wag oraz zależność samych wag od kontekstu – za pomocą funkcji bazowych i odpowiadających im współczynników.

6.3. Przetwarzanie intensywnych strumieni danych przez sieci neuronowe

W dzisiejszym świecie konieczność przetwarzania intensywnych strumieni danych staje się coraz bardziej znacząca. W bardziej ogólnym sensie, spotykamy się z zalewem informacji na każdy temat, spowodowanym szybszym tempem życia, rozpowszechnieniem metod cyfrowej rejestracji sygnałów (dźwięku, obrazu). Sprawia to, że coraz częściej komputery muszą przetwarzać (co dla nas oznacza *analizować*, a nie tylko przesyłać i zapisywać) intensywne strumienie danych,

przez które rozumiemy ciągły, potencjalnie nieskończenie długotrwały dopływ informacji wielowymiarowych (wektorowych, macierzowych, 3D). Mogą być one generowane na przykład przez radioteleskopy kosmiczne analizujące napływające sygnały o szerokim spektrum, czy przez systemy płatności kartami kredytowymi. Do obróbki takich danych potrzeba algorytmów szczególnie wydajnych. Jeśli algorytmami tymi będą sieci neuronowe, musimy poszukiwać sieci niewielkich rozmiarów, działających na tyle szybko, aby możliwe było przetwarzanie informacji (wyznaczanie odpowiedzi nauczonych sieci) w czasie rzeczywistym (*on-line*), a także szybkie bieżące douczanie na ciągle napływających danych.

Tradycyjne sieci neuronowe osiągają te cele dzięki równoległości przetwarzania danych. W dalszej części pokażemy, że kontekstowe sieci neuronowe zawdzięczają dużą wydajność w rozwiązywaniu pewnych klas zadań nie tylko tej naturalnej równoległości, ale również lepszemu dopasowaniu swej struktury do charakteru rozwiązywanych problemów i przetwarzanych danych. Pokażemy, że – mimo pozornego skomplikowania modelu sieci – staje się on bardziej przejrzysty dla zadań o charakterze kontekstowym, co umożliwia używanie mniejszych rozmiarami sieci do rozwiązywania tej klasy zadań. Użycie zaś odpowiednich algorytmów uczenia sprawia, że uczenie i używanie tych sieci ma złożoność obliczeniową zbliżoną do sieci tradycyjnych o tym samym rozmiarze, ale o mniejszych zdolnościach przetwarzania danych kontekstowych.

6.4. Model kontekstowego neuronu

W tym podrozdziale przedstawiony zostanie model neuronu kontekstowego, zostaną także ogólnie przeanalizowane różnice w jego budowie i sposobie działania w stosunku do modelu tradycyjnego. Model ten został wprowadzony w [11], rozwinięty w [5] i szerzej przedstawiony w [6].

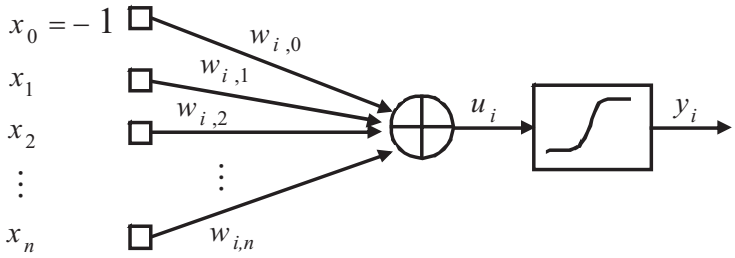
Rozważmy model tradycyjnego neuronu pokazany na rysunku 6.3. Niech będzie to i -ty neuron w warstwie sieci, o wagach $w_{i,j}$ prowadzących do wejść tradycyjnych x_j oraz do wejścia progowego (ang. *bias*) oznaczonego jako x_0, przy czym $i = 1, 2, ..., M$, $j = 1, 2, ..., n$.

Pobudzenie neuronu u_i dane jest wzorem

$$u_i = \sum_{j=0}^{n} w_{i,j} x_j = \mathbf{w}_k^T \mathbf{x}, \tag{6.1}$$

natomiast jego wyjście y_i określa zależność

$$y_i = f(u_i), \tag{6.2}$$



Rysunek 6.3. Model neuronu tradycyjnego

gdzie $\mathbf{w}_i = [w_{i,0}, w_{i,1}, ..., w_{i,n}]^T$ jest wektorem kolumnowym wszystkich wag neuronu, $\mathbf{x} = [x_0, x_1, ..., x_n]^T$ kolumnowym wektorem jego wejść (z wejściem progowym włącznie).

Sieci wielowarstwowe budowane są z kolejnych warstw neuronów, w których wszystkie wyjścia neuronów poprzedniej warstwy stanowią wejścia dla neuronów następnej warstwy. Korzystając z przedstawionego zapisu wektorowego oraz grupując wagi poszczególnych neuronów w ich wektorach wagowych, natomiast wektory wagowe neuronów danej warstwy w macierz wag, można zapisać zwięźle wzór na pobudzenie warstwy sieci

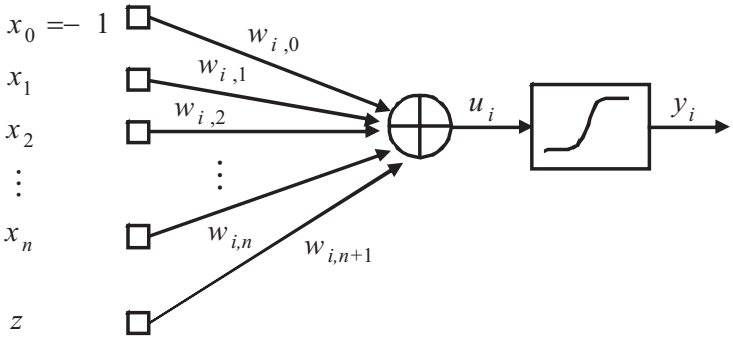
$$\mathbf{u} = \mathbf{w}^T \mathbf{x}, \tag{6.3}$$

gdzie $\mathbf{u} = [u_1, u_2, ..., u_M]^T$ jest wektorem kolumnowym pobudzeń wszystkich M neuronów w warstwie sieci, $\mathbf{w}$ zaś jest macierzą wag warstwy sieci, złożoną z wektorów wag poszczególnych neuronów

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_M \end{bmatrix} = \begin{bmatrix} w_{1,0} & w_{2,0} & \cdots & w_{M,0} \\ w_{1,1} & w_{2,1} & \cdots & w_{M,1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,n} & w_{2,n} & \cdots & w_{M,n} \end{bmatrix}. \tag{6.4}$$

Przedstawiona sieć tradycyjna nie używa informacji kontekstowej – przetwarza jedynie podstawowe cechy badanego obiektu. Nawiązując do przedstawionego wcześniej przykładu zastosowania medycznego, na wejścia sieci zostanie podany przebieg czasowy sygnału z pewnego badania pacjenta (poddany już wstępnej obróbce statystycznej), np. przebieg EKG, natomiast inne dane o pacjencie niosące informację kontekstową, np. jego wiek – będą przez tę sieć pomijane.

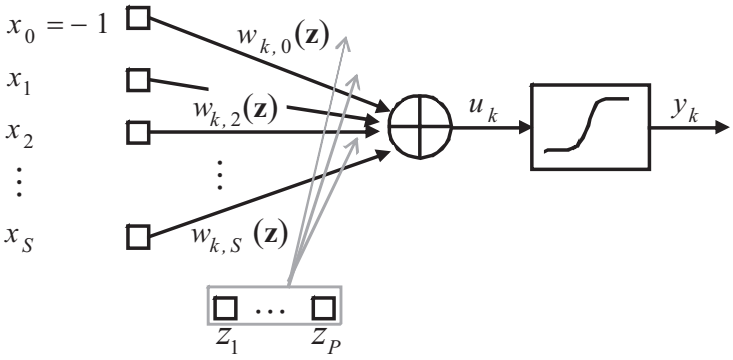
Załóżmy teraz, że informacja kontekstowa zostanie podana na wejście sieci tradycyjnej w postaci jednej cechy z o charakterze kontekstowym. Jedynym



Rysunek 6.4. Neuron tradycyjny wykorzystujący dane kontekstowe

sposobem, w jaki sieć tradycyjna może wykorzystać te dane, jest rozszerzenie wektora wejściowego neuronów pierwszej warstwy sieci o kolejne wejście, co dla pojedynczego neuronu pokazano na rysunku 6.4. Dokładniejsza analiza sposobu działania sieci tradycyjnych oraz kontekstowych, w przestrzeni sygnałów wejściowych zawierającej kontekst, zostanie przedstawiona w następnym podrozdziale. W tym miejscu zauważmy, że neuron tradycyjny przetwarza cechy podstawowe oraz kontekstowe analizowanego problemu w ten sam sposób. Podejście to jest niezgodne z ludzką intuicją podpowiadającą na przykład, że podczas podejmowania decyzji o stanie zdrowia pacjenta, dane dotyczące jego przebiegu EKG oraz wieku nie niosą równorzędnej informacji. Intuicja podpowiada również, że sposób analizowania przebiegu EKG powinien być uzależniony od wieku pacjenta. Jak zobaczymy w dalszej części rozdziału, kontekstowe sieci neuronowe uzależniają przetwarzanie danych podstawowych od informacji kontekstowych.

Model neuronu kontekstowego w najbardziej ogólnej postaci przedstawiony jest na rysunku 6.5. Jedyną, ale też najważniejszą, różnicą w stosunku do neuronu tradycyjnego jest fakt, że wagi neuronu nie są ustalonymi liczbami, lecz funkcjami pewnych zmiennych. Mamy tu przy tym na myśli sieć już nauczoną – w sieci tradycyjnej wagi podlegają zmianom w procesie uczenia, potem jednak są już stałe. Wagi neuronu kontekstowego nie są liczbami, lecz są zależne funkcyjnie od wartości wejść kontekstowych. Modelowanie tej zależności może zostać przeprowadzone w dowolny sposób. W przedstawionym wcześniej przykładzie kontekstowego wyznaczania kinematyki odwrotnej ramienia robota z pracy [14] wykorzystywana była do tego celu kolejna sieć neuronowa, ucząca się zależności wag sieci nadrzędnej od kontekstu. W niniejszej pracy przedstawimy model średniokwadratowy modelowania tej zależności, wykorzystujący wektor funkcji bazowych oraz iloczyn Kroneckera wejść tradycyjnych z wektorem wartości funkcji bazowych. Omówione



Rysunek 6.5. Model neuronu kontekstowego

zostaną również zalety takiego rozwiązania, w szczególności prostota zapisu oraz jego komplementarność względem zapisu modelu tradycyjnego sieci.

Rozważmy neuron kontekstowy z rysunku 6.5. Ma on n wejść podstawowych oraz p wejść kontekstowych. W modelu średniokwadratowym waga i-tego neuronu do wejścia podstawowego o numerze j, oznaczona jako $w_{i,j}$, zależy od wektora zmiennych kontekstowych $\mathbf{z}$ zgodnie ze wzorem

$$w_{i,j}(\mathbf{z}) = \sum_{k=1}^{r} a_{i,j,k} v_k(\mathbf{z}) = \mathbf{a}_{i,j}^T \mathbf{v}(\mathbf{z}), \tag{6.5}$$

gdzie $\mathbf{v}(\mathbf{z})$ jest wektorem r niezależnych funkcji bazowych, wspólnym dla całej sieci, natomiast $\mathbf{a}_{i,j}$ jest wektorem współczynników dla wagi $w_{i,j}$. Każdej wadze sieci tradycyjnej odpowiada teraz wektor współczynników, przy czym ich liczba nie jest równa liczbie p wejść kontekstowych, lecz wybranej przez nas liczbie r funkcji bazowych. Pobudzenie i-tego neuronu u_i oraz jego wyjście y_i są dane tymi samymi wzorami, co dla neuronu tradycyjnego, czyli (6.1) oraz (6.2), z zastrzeżeniem jednak, że macierz wag jest teraz funkcyjnie zależna od wektora kontekstu.

Sygnał wyjściowy neuronu, na którego wejścia podstawowe podano wektor $\mathbf{x}$, a na wejścia kontekstowe wektor $\mathbf{z}$, można wyznaczyć w czterech krokach:

1. obliczenie wartości wektora funkcji bazowych $\mathbf{v}(\mathbf{z})$ dla bieżącego kontekstu określonego wektorem $\mathbf{z}$ (w sieciach wielowarstwowych wystarczy to zrobić raz dla wszystkich neuronów sieci),
2. obliczenie wartości wektora wag neuronu $\mathbf{w}_i(\mathbf{z})$ w bieżącym kontekście $\mathbf{z}$,
3. obliczenie pobudzenia neuronu u_i,
4. obliczenie wyjścia neuronu y_i.

Jak widać, schemat ten jest analogiczny do schematu postępowania dla neuronu tradycyjnego, jednak uzupełniony o dwa pierwsze kroki. Obliczenie wektora pobudzeń $\mathbf{u}$ jednej warstwy sieci musi zostać poprzedzone obliczeniem wartości wektora funkcji bazowych $\mathbf{v}(\mathbf{z})$ w bieżącym kontekście $\mathbf{z}$ oraz wartości wag $\mathbf{w}(\mathbf{z})$ wszystkich neuronów danej warstwy w danym kontekście. Obliczenie wektora pobudzeń wszystkich neuronów w warstwie odbywa się już analogicznie do sieci tradycyjnej, według wzoru

$$\mathbf{u} = \mathbf{w}^T(\mathbf{z})\,\mathbf{x}. \tag{6.6}$$

Przedstawiony zostanie teraz sposób zapisu wektora współczynników pojedynczego neuronu oraz macierzy współczynników całej warstwy neuronów, pozwalający uprościć powyższy schemat przez wyeliminowanie z niego etapu obliczania wartości wag w bieżącym kontekście.

Parametrami neuronu kontekstowego są współczynniki a przy funkcjach bazowych, modelujące zależność wag od kontekstu. Wadze $w_{i,j}$ neuronu tradycyjnego, czyli wadze i-tego neuronu prowadzącej do wejścia j, odpowiada w neuronie kontekstowym wektor współczynników

$$\mathbf{a}_{i,j} = \begin{bmatrix} a_{i,j,1} \\ a_{i,j,2} \\ \vdots \\ a_{i,j,r} \end{bmatrix}. \tag{6.7}$$

Wektory współczynników wag i-tego neuronu możemy „złożyć jeden na drugim" na wzór wektora wag neuronu, zastępując w nim każdą wagę $w_{i,j}$ odpowiadającym jej wektorem współczynników $\mathbf{a}_{i,j}$, w następujący sposób:

$$\mathbf{a}_i = \begin{bmatrix} \mathbf{a}_{i,0} \\ \mathbf{a}_{i,1} \\ \vdots \\ \mathbf{a}_{i,n} \end{bmatrix}, \tag{6.8}$$

otrzymując wektor współczynników o długości $(n+1)r \times 1$.

Wzorując się na macierzy wag dla warstwy liczącej M neuronów tradycyjnych, tworzymy analogicznie macierz współczynników dla warstwy M neuronów kontekstowych, umieszczając obok siebie wektory współczynników poszczególnych neuronów:

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_K \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{1,0} & \mathbf{a}_{2,0} & \cdots & \mathbf{a}_{M,0} \\ \mathbf{a}_{1,1} & \mathbf{a}_{2,1} & \cdots & \mathbf{a}_{M,1} \\ \vdots & \vdots & & \vdots \\ \mathbf{a}_{1,n} & \mathbf{a}_{2,n} & \cdots & \mathbf{a}_{M,n} \end{bmatrix}. \tag{6.9}$$

Tak zapisana macierz współczynników ma wymiar $(n+1)r \times M$. Wektor pobudzeń wszystkich neuronów w warstwie można teraz zapisać w postaci

$$\mathbf{u} = \mathbf{w}^T(\mathbf{z})\,\mathbf{x} = \mathbf{a}^T\left[\mathbf{x} \otimes \mathbf{v}(\mathbf{z})\right]. \tag{6.10}$$

Zapis ten pozwala na skrócenie obliczania wyjścia warstwy neuronów przez wyeliminowanie etapu obliczania wartości wag neuronów w danym kontekście. Jednak główną jego zaletą jest uproszczenie zapisu gradientu funkcji błędu względem macierzy współczynników, używanego nie tylko w najprostszej metodzie najszybszego spadku, lecz także w innych metodach uczenia sieci opartych na obliczaniu gradientu.

6.5. Porównanie sposobu działania sieci tradycyjnych i kontekstowych

W niniejszym podrozdziale, na przykładzie prostych problemów klasyfikacji wzorców, porównamy działanie i zdolności rozdzielcze neuronu tradycyjnego oraz kontekstowego w wielowymiarowej przestrzeni wejściowej zawierającej wejścia podstawowe oraz kontekstowe.

W zadaniach klasyfikacji neuron tradycyjny zachowuje się jak dyskryminator liniowy. Neuron o dwóch wejściach potrafi zaklasyfikować sygnały wejściowe do dwóch klas, oddzielając punkty na płaszczyźnie sygnałów wejściowych linią prostą. Przy większej liczbie wejść neuron dzieli punkty na dwie klasy w przestrzeni sygnałów wejściowych o odpowiednio większym wymiarze za pomocą hiperpłaszczyzny decyzyjnej. Przestrzeń parametrów neuronu odpowiada przy tym dokładnie przestrzeni jego sygnałów wejściowych (jest zawsze o 1 większa od liczby wejść, gdyż obejmuje wagę prowadzącą do ukrytego wejścia progowego, tzw. biasu).

Dodanie do wektora wejść tradycyjnego neuronu jednego wejścia z informacją kontekstową powoduje proste zwiększenie wymiaru przestrzeni sygnałów wejściowych o jeden i takie samo rozszerzenie jego przestrzeni parametrów. Nie zmienia przy tym jego sposobu działania – neuron dalej będzie potrafił rozdzielać punkty za pomocą hiperpłaszczyzny, przy czym kontekst będzie po prostu jednym z wymiarów przestrzeni wejściowej.

W neuronie kontekstowym przestrzeń parametrów neuronu nie odpowiada dokładnie przestrzeni sygnałów wejściowych. Wymiar przestrzeni parametrów zależy od liczby wejść kontekstowych nie bezpośrednio, lecz przez liczbę funkcji bazowych używanych do modelowania wag. Funkcje bazowe są funkcjami wejść kontekstowych. Samo dodanie wejścia kontekstowego nie wpływa jeszcze w żaden

sposób na własności neuronu. Po dodaniu jednego wejścia kontekstowego należy rozszerzyć wektor funkcji bazowych o co najmniej jedną funkcję korzystającą z tego wejścia, natomiast wszystkie wektory współczynników przybliżających wagi o jeden współczynnik. Po tych zabiegach przestrzeń parametrów neuronu powiększy się nie o 1, jak w neuronie tradycyjnym, lecz o $n + 1$, czyli o liczbę wag neuronu. Możemy przy tym zwiększyć liczbę funkcji bazowych o więcej niż jedną, pamiętając by były one liniowo niezależne. Możemy więc sami wpływać na zdolności rozdzielcze sieci.

Dzięki temu, że przestrzeń parametrów neuronu kontekstowego jest większa niż przestrzeń sygnałów wejściowych (przez co w pewnym sensie przypomina on neuron z połączeniami funkcyjnymi Pao [10]), granice decyzyjne generowane przez neuron kontekstowy nie są hiperpłaszczyznami, lecz hiperpowierzchniami w przestrzeni sygnałów wejściowych. Pozostają przy tym hiperpłaszczyznami w przestrzeni parametrów neuronu oraz w przekroju przestrzeni wejściowej, oznaczającym jeden kontekst, co zostanie pokazane w dalszej części rozdziału.

Podane rozważania wiążą się ściśle z teorią uczenia maszynowego, a w szczególności z pojęciem wymiaru Vapnika–Chervonenkisa. Wielkością tą mierzy się zdolności rozdzielcze maszyny uczącej, pozwala ona również oszacować liczbę przykładów niezbędnych do nauczenia jej do wymaganego poziomu błędu, lub – przy określonej liczbie przykładów – błąd, jaki sieć jest w stanie osiągnąć podczas uczenia.

Wymiar Vapnika–Chervonenkisa neuronu tradycyjnego oraz sieci tradycyjnej jest mocno związany z przestrzenią jego parametrów. Dla pojedynczego neuronu wynosi on $n + 1$, gdzie n jest liczbą wejść, czyli jest dokładnie równy liczbie parametrów (wag) neuronu. Jeśli neuron tradycyjny korzysta z informacji kontekstowej, podanej na p wejściach tradycyjnych, jego wymiar VC-dim będzie wynosił $n + p + 1$.

Jak dowiedziono w [5] oraz [4], wymiar Vaplika-Chervonenkisa dla pojedynczego neuronu kontekstowego jest równy $r\,(n + 1)$, czyli również liczbie parametrów neuronu. Jednak w tym przypadku liczba parametrów neuronu nie jest równa liczbie jego wejść (łącznie podstawowych i kontekstowych), czego konsekwencje zostaną teraz pokazane.

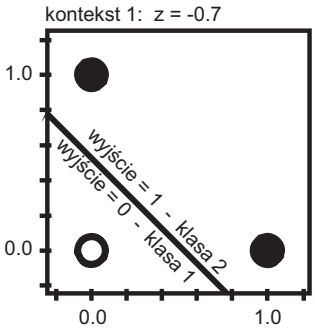
Aby neuron kontekstowy mógł efektywnie wykorzystać informację kontekstową podaną mu na p wejściach, muszą być spełnione dwa warunki:

— $r > 1$ – gdyż dla $r = 1$ neuron kontekstowy ma własności neuronu tradycyjnego,

— $r \geqslant p$ – aby użyć informacji ze wszystkich wejść kontekstowych.

Przy tych założeniach możemy zauważyć, że

$$r \cdot (n + 1) > n + p + 1, \tag{6.11}$$



Rysunek 6.6. Problem OR – granica decyzyjna neuronu tradycyjnego (punkt biały – klasa 0, punkty czarne – klasa 1)

co oznacza, że wymiar VC-dim neuronu kontekstowego korzystającego z n wejść podstawowych i p wejść kontekstowych, będzie zawsze większy od wymiaru VC-dim neuronu tradycyjnego, przetwarzającego te same informacje, ale podane na $n + p$ wejść podstawowych.

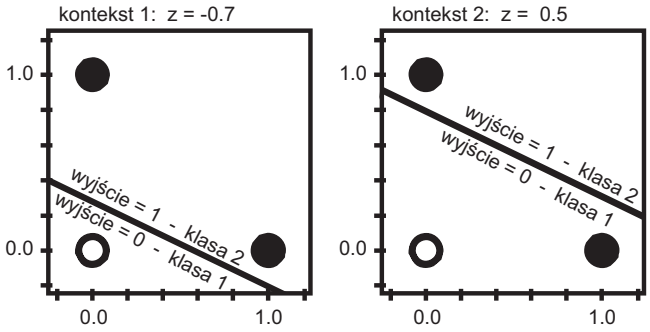
Dla tradycyjnej sieci wielowarstwowej oszacowanie wymiaru VC-dim jest rzędu $O\ (W^2)$, gdzie W jest liczbą wag całej sieci. Natomiast wymiar VC-dim wielowarstwowej sieci złożonej z neuronów kontekstowych jest rzędu $O\ (A^2)$, gdzie A jest liczbą wszystkich współczynników sieci. Zdolności rozdzielcze sieci kontekstowych zależą więc od liczby parametrów tak samo, jak w sieciach tradycyjnych. Jednak zależność ta inaczej przekłada się na ich zdolności rozdzielcze w przestrzeni sygnałów wejściowych.

Podane rozważania doskonale zilustrują dwa przykłady prostej klasyfikacji. Pojedyncze neurony oraz sieci tradycyjne i kontekstowe zostaną w nich zastosowane do rozwiązywania problemów OR oraz XOR rozmieszczonych w kilku kontekstach. Pierwszy przykład pokazuje możliwości kształtowania granic decyzyjnych jednego neuronu w przestrzeni wejściowej w wielu kontekstach, drugi zaś ilustruje dopasowanie modelu wielowarstwowych sieci kontekstowych do problemów o charakterze kontekstowym.

Klasyczny problem OR polega na rozdzieleniu trzech punktów na płaszczyźnie do dwóch klas. Punkty te mogą być zebrane jako macierz wzorców uczących dla sieci neuronowej, w której przykłady umieszczone są w kolejnych kolumnach, natomiast wiersze odpowiadają dwóm wejściom podstawowym sieci tradycyjnej

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Punkty te powinny być rozdzielone do dwóch klas oznaczonych 0 i 1 tak, jakby



Rysunek 6.7. Problem OR w dwóch kontekstach – granice decyzyjne neuronu tradycyjnego (zadanie rozwiązane w obu kontekstach)

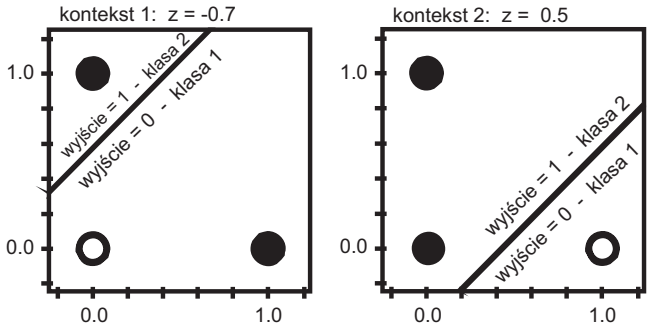
przeszły przez bramkę OR. Wektor żądanych wyjść sieci dla kolejnych przykładów dany jest więc jako

$$T = \left[\begin{array}{ccc} 0 & 1 & 1 \end{array} \right].$$

Zadanie to zilustrowano na rysunku 6.6, na którym, oprócz punktów wzorcowych zaklasyfikowanych do dwóch klas, pokazano również granicę decyzyjną neuronu tradycyjnego rozwiązującego to zadanie. Neuron ten, mając 3 wagi, a tym samym wymiar VC-dim równy 3, może przeprowadzić wszystkie dychotomie 3 punktów na płaszczyźnie. Oznacza to, że możemy tak dobrać jego wagi, aby te 3 punkty rozdzielić na dwie klasy na wszystkie z 8 możliwości.

Dodajmy trzecie wejście do neuronu, oznaczając je przez z, i potraktujmy jako kontekst, w którym znajdują się rozdzielane punkty. Przestrzeń sygnałów wejściowych neuronu stała się teraz trójwymiarowa. Jeśli przypiszemy trzem pierwszym punktom tę samą wartość zmiennej kontekstowej, będziemy mogli powiedzieć, że leżą one w tym samym kontekście, czyli w pewnym przekroju przestrzeni wejść wzdłuż osi z. Przestrzeń wag, czyli parametrów tradycyjnego neuronu zwiększyła się teraz wraz z przestrzenią wejściową do 3, natomiast jego wymiar VC-dim wzrósł do 4. Oznacza to, że w trójwymiarowej przestrzeni wejściowej neuron może teraz rozdzielić 4 punkty na wszystkie możliwe 16 sposobów, prowadząc w tej przestrzeni płaszczyznę decyzyjną. W konkretnym kontekście (czyli przekroju przestrzeni wejściowej wzdłuż osi z, w tym przypadku jest to płaszczyzna) nadal może jednak rozdzielić tylko 3 punkty, prowadząc w nim linię decyzyjną.

Podczas rozszerzenia przestrzeni wejść neuron otrzymał dodatkowy parametr, dzięki któremu mając ustaloną granicę decyzyjną w pewnym kontekście (np. zgodnie z rys. 6.6), może dopasować swoją płaszczyznę decyzyjną w innym kontekście. W drugim przekroju przestrzeni wejściowej linia decyzyjna musi być jednak rów-

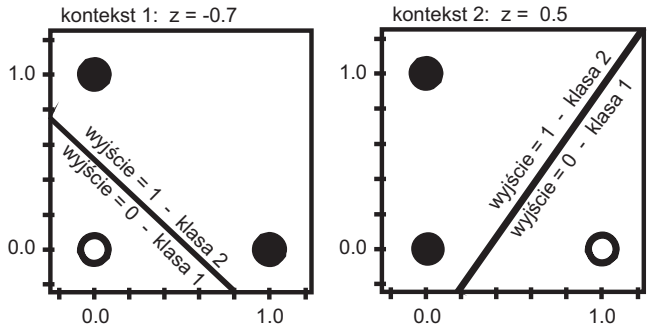


Rysunek 6.8. Problem OR w dwóch kontekstach – granice decyzyjne neuronu tradycyjnego (zadanie niemożliwe do rozwiązania w pierwszym kontekście)

noległa do tej z pierwszego przekroju. Umożliwi ona dokonanie wszystkich możliwych dychotomii trzech punktów z pierwszego kontekstu oraz jednego dodatkowego punktu w drugim kontekście. Będzie można również rozdzielić w drugim kontekście takie punkty, dla których odpowiednia jest równoległa granica decyzyjna (jak na rysunku 6.7), ale nie będzie możliwe poprowadzenie linii decyzyjnej pod innym kątem, koniecznej na przykład do rozdzielenia punktów z rysunku 6.8. Nie podważa to oczywiście zdolności neuronu tradycyjnego do rozdzielenia 16 punktów na wszystkie możliwe sposoby. Jednak punkty te muszą być umieszczone w przynajmniej trzech przekrojach przestrzeni wejściowej wzdłuż osi z.

Jak już zauważono wcześniej, w pojedynczym kontekście neuron kontekstowy ma własności neuronu tradycyjnego. Neuron kontekstowy o dwóch wejściach podstawowych oraz jednej stałej funkcji bazowej jest więc w stanie wyznaczyć linię graniczną dla dwóch klas punktów (jak na rysunku 6.6). Jeśli jednak dodamy jedno wejście kontekstowe z oraz jedną funkcję bazową, korzystającą z tego wejścia, to przestrzeń wejściowa neuronu zwiększy się o 1, natomiast przestrzeń jego parametrów – o liczbę wag, czyli $n + 1$. Jeśli wejście z odłożymy na jednej z osi przestrzeni wejściowej, to w jednym przekroju dla $z = z_1$ neuron kontekstowy będzie mógł wyznaczyć jedną liniową granicę decyzyjną, natomiast w drugim przekroju dla $z = z_2$ inną granicę liniową, całkowicie niezależną od tej pierwszej.

Przypomnijmy, że neuron tradycyjny był w stanie poprowadzić w drugim przekroju jedynie granicę równoległą do granicy wcześniej ustalonej w pierwszym kontekście. Natomiast neuron kontekstowy w każdym z takich przekrojów będzie potrafił rozdzielić na wszystkie możliwe sposoby nie więcej niż 3 punkty. Przy tym każda dodatkowa funkcja bazowa dodaje możliwość poprowadzenia liniowej granicy decyzyjnej w kolejnym kontekście, zupełnie niezależnie od granic decyzyj-



Rysunek 6.9. Problem OR w dwóch kontekstach – granice decyzyjne neuronu kontekstowego

nych ustalonych w innych kontekstach. Przykład takich niezależnych granic dla punktów rozmieszczonych w dwóch kontekstach przedstawiono na rysunku 6.9.

Następnym przykładem jest problem XOR rozmieszczony w kilku kontekstach. Choć w klasycznej postaci zadanie to jest znane od lat, po rozszerzeniu o wymiar kontekstowy doskonale ilustruje zasady działania tradycyjnych oraz kontekstowych sieci wielowarstwowych.

Klasyczny problem XOR polega na rozdzieleniu czterech punktów na płaszczyźnie do dwóch klas. Punkty te mogą być zebrane w macierzy wzorców uczących dla sieci neuronowej, analogicznie do poprzedniego przykładu:

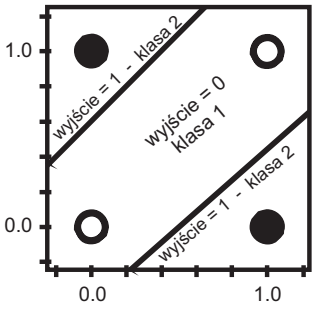
$$P = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

Punkty te powinny być rozdzielone do dwóch klas oznaczonych etykietami 0 i 1 tak, jakby przeszły przez bramkę XOR. Wektor żądanych wyjść sieci dla kolejnych przykładów dany jest więc jako

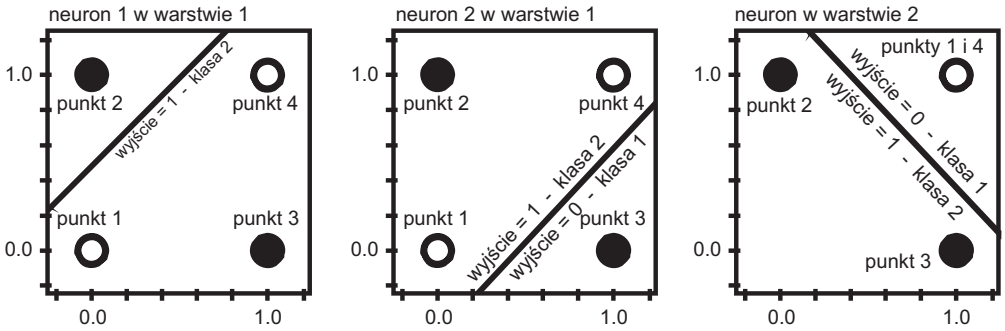
$$T = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}.$$

Zadanie to zilustrowano na rysunku 6.10, na którym, oprócz punktów wzorcowych zaklasyfikowanych do dwóch klas, pokazano również granice decyzyjne tradycyjnej sieci dwuwarstwowej.

Jak wiadomo, problem XOR jest najmniejszym problemem nieseparowalnym liniowo, czyli niemożliwym do rozdzielenia za pomocą pojedynczej linii prostej. Dlatego też nie może być rozwiązany przez pojedynczy neuron, gdyż ten jest w stanie wyznaczyć jedynie liniową granicę decyzyjną. Najmniejszą siecią tradycyjną, zdolną rozwiązać to zadanie, jest sieć o strukturze 2-2-1, czyli o dwóch



Rysunek 6.10. Klasyczny przykład XOR i granice decyzyjne sieci tradycyjnej (punkty białe – klasa 0, czarne – klasa 1)



Rysunek 6.11. Klasyczny przykład XOR – granice decyzyjne dwóch neuronów warstwy pierwszej oraz jednego neuronu warstwy drugiej

wejściach, dwóch neuronach w warstwie ukrytej i jednym w warstwie wyjściowej. Jest to sieć o wymiarze VC-dim równym 9 (gdyż zawiera ona 9 wag), a więc dużo większym niż potrzebny do rozdzielenia czterech punktów. Jak pokazano na rysunku 6.11, sieć dwuwarstwowa rozwiązuje ten problem w dwóch etapach. Każdy neuron warstwy pierwszej klasyfikuje 3 punkty do jednej klasy oraz 1 punkt do drugiej, przy czym ich granice decyzyjne są równoległe i przesunięte. Tak przetworzony przez warstwę pierwszą problem staje się liniowo separowalny i może zostać ostatecznie rozwiązany za pomocą jednego neuronu drugiej warstwy sieci.

Zanim przejdziemy do dalszej analizy przykładu XOR w wielu kontekstach, zauważmy, że model neuronu kontekstowego jest uogólnieniem modelu tradycyjnego. Pomimo, że kontekst jest w naszym modelu zmienną ciągłą, to dane podstawowe o tych samych wartościach wektora wejść kontekstowych (a tym samym o jednakowych wartościach wektora funkcji bazowych) możemy uważać za pocho-

dzące z jednego kontekstu. Jeśli wektor wejść kontekstowych składa się z jednej zmiennej, możemy powiedzieć, że dane te leżą w tym samym przekroju przestrzeni sygnałów wejściowych. W tak rozumianym pojedynczym kontekście neuron kontekstowy jest dokładnym odpowiednikiem neuronu tradycyjnego – ma takie same możliwości przetwarzania danych podstawowych jak neuron tradycyjny o tej samej liczbie wag nieużywający informacji o kontekście. Oba rodzaje neuronów są więc w stanie wyznaczyć liniową granicę decyzyjną między danymi wejściowymi znajdującymi się w jednym kontekście. W sieciach tradycyjnych nieliniowe granice decyzyjne modelowane są dzięki użyciu wielu warstw. Sieć dwuwarstwowa potrafi w drugiej warstwie „złożyć" nieliniową granicę decyzyjną z granic liniowych dostarczonych przez odpowiednią liczbę neuronów warstwy pierwszej. Dokładnie tak samo kontekstowa sieć wielowarstwowa utworzy nieliniową granicę decyzyjną w pojedynczym kontekście. Inaczej jednak zamodeluje zmianę tej nieliniowej granicy decyzyjnej pomiędzy kontekstami.

Sieci kontekstowe są uogólnieniem modelu tradycyjnego również dlatego, że po zastosowaniu wektora funkcji bazowych zawierającego jedną funkcję bazową, sieć staje się dokładnym odpowiednikiem sieci tradycyjnej o tej samej strukturze (nie korzystającej z wejść kontekstowych). Analizowany wcześniej standardowy problem XOR może więc rozwiązać np. sieć kontekstowa o strukturze 2-2-1 korzystająca z wektora funkcji bazowych $\mathbf{v}(\mathbf{z}) = [1]$, czyli „udająca" sieć tradycyjną. Przy pojedynczej funkcji bazowej wartości wejść kontekstowych dla poszczególnych przykładów z ciągu uczącego nie mają znaczenia.

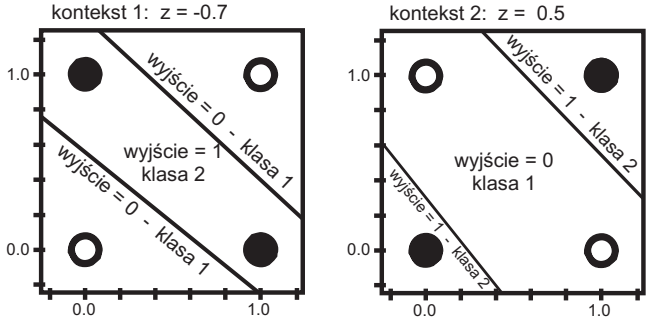
Zdefiniujmy teraz problem XOR w dwóch kontekstach. Ciąg uczący dla sieci neuronowej dany jest za pomocą zestawu przykładów uczących, czyli trójek wektorów – wejść podstawowych $\mathbf{x}_k$, kontekstowych $\mathbf{z}_k$ oraz żądanych wyjść $\mathbf{y}_k$, zdefiniowanych następująco

$$\{(\mathbf{x}_k, \mathbf{z}_k, \mathbf{y}_k)\}_{k=1}^{N} \ , \tag{6.12}$$

gdzie:

$$\begin{aligned}
\{\mathbf{x}_k\}_{k=1}^{N} &= \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \\
\{\mathbf{z}_k\}_{k=1}^{N} &= \begin{bmatrix} -0.7 & -0.7 & -0.7 & -0.7 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}, \\
\{\mathbf{y}_k\}_{k=1}^{N} &= \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix},
\end{aligned}$$

w których przykłady umieszczone są w kolejnych kolumnach: pierwsze cztery z nich umieszczone są w jednym kontekście, cztery następne zaś w drugim (wartości zmiennej kontekstowej dla obu kontekstów zostały dobrane arbitralnie). Jak widać, klasyczny problem XOR został rozszerzony o kolejne 4 punkty w drugim



Rysunek 6.12. Przykład XOR w dwóch kontekstach
– granice decyzyjne sieci kontekstowej o strukturze 2-2-1:2 (18 współczynników)
w poszczególnych kontekstach

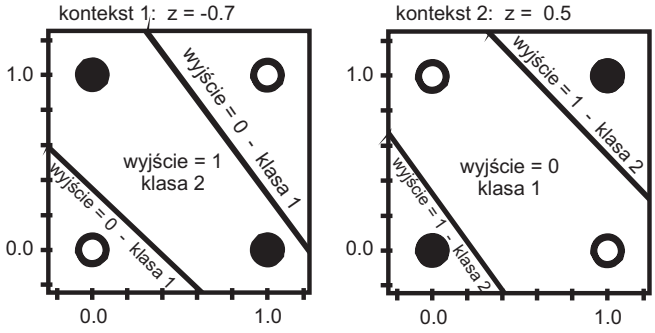
kontekście, a ich klasyfikacja ma być odwróceniem klasyfikacji czterech punktów z kontekstu pierwszego. Wymaga więc ustalenia w obu kontekstach zupełnie innych granic decyzyjnych. Zadanie to przedstawione jest na rysunku 6.12 wraz z granicami decyzyjnymi, jakie w obu kontekstach zapewnia sieć kontekstowa. Jest to przykład problemu zgodnego z ideą modelowania kontekstowego – złożoność problemu nie zmienia się wraz z kontekstem, nie zmienia się więc również postać optymalnego rozwiązania, a jedynie jego parametry – w tym przypadku wraz ze zmianą kontekstu zmienia się nie kształt, lecz tylko położenie granic decyzyjnych. Skoro znamy sieć o optymalnej strukturze i wartościach parametrów, zdolną rozwiązać problem w jednym kontekście, wystarczy zmienić jej parametry na optymalne dla innego kontekstu bez zmieniania jej struktury.

Najmniejszą siecią kontekstową zdolną rozwiązać zadanie XOR niezależnie w dwóch kontekstach jest sieć o strukturze 2-2-1 (czyli takiej jak sieć tradycyjna dla problemu XOR w jednym kontekście), wykorzystująca dwie funkcje w wektorze funkcji bazowych. Taką strukturę sieci będziemy zapisywać jako 2-2-1:2. Wektor funkcji bazowych może być na przykład postaci

$$\mathbf{v}\left(\mathbf{z}\right) = \begin{bmatrix} 1 \\ z \end{bmatrix}.$$

Sieć taka ma po 2 współczynniki przybliżające każdą wagę, w sumie więc rozmiar sieci to 18 współczynników.

Ze względu na nadmiarowość podstawowej struktury tradycyjnej dla problemu XOR w jednym kontekście (9 wag do rozdzielenia 4 punktów), do rozwiązania problemu XOR w dwóch kontekstach wystarczy sieć tradycyjna o strukturze 3-2-1 (11 wagach), a więc również o 2 neuronach w warstwie ukrytej, przyjmująca

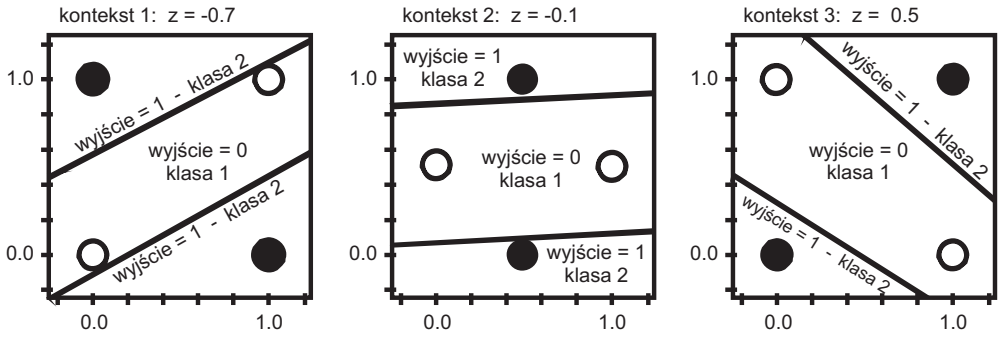


Rysunek 6.13. Przykład XOR w dwóch kontekstach – granice decyzyjne sieci tradycyjnej o strukturze 3-2-1 (11 wag) w poszczególnych kontekstach

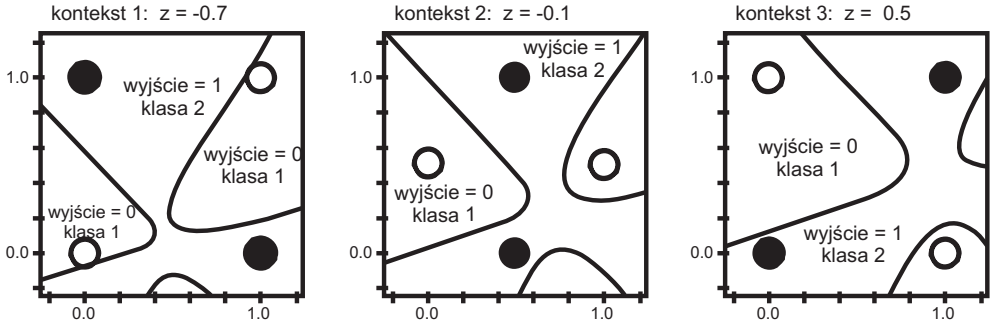
po prostu wartość kontekstu na dodatkowe trzecie wejście podstawowe. Granice decyzyjne sieci tradycyjnej przedstawiono na rysunku 6.13.

Choć wyglądają podobnie do granic decyzyjnych sieci kontekstowej, ich mechanizm powstawania jest inny. Możemy go zilustrować, wprowadzając do zadania XOR w dwóch kontekstach wymóg, aby granice decyzyjne zmieniały się w sposób ciągły pomiędzy dwoma kontekstami. Do ciągu uczącego wprowadzimy więc cztery dodatkowe punkty w kontekście przejściowym obrócone w stosunku do obu pierwotnych kontekstów o 45 stopni. Punkty te pokazane są na rysunku 6.14, na którym umieszczono również granice decyzyjne sieci kontekstowej o strukturze takiej jak poprzednio, czyli 2-2-1:2 (18 współczynników). Widać, że granice w kolejnych kontekstach są przesuniętymi granicami z pierwszego kontekstu, ustalonymi dla pojedynczego problemu XOR. We wszystkich kontekstach granice te mają ten sam kształt. Generowane są one przez 9 wag neuronu, które zależą od wartości wejścia kontekstowego. Zależności te pokazano na rysunku 6.17.

Sieć tradycyjna o strukturze 3-2-1, która poprzednio rozwiązywała zadanie XOR w dwóch kontekstach, nie jest już w stanie rozwiązać obecnego zadania. Dla tej struktury sieci należy to zadanie potraktować jako problem XOR w trzech niezależnych kontekstach, w którym rozdzielić należy 12 punktów, a na to rozmiar tej sieci (11 wag) jest już za mały. Aby uzyskać sieć zdolną do rozwiązania tego problemu, należy zwiększyć liczbę jej neuronów ukrytych do 3, a tym samym wymiar VC-dim sieci do 16. Granice decyzyjne takiej sieci pokazane są na rysunku 6.15 we wszystkich trzech interesujących nas przekrojach przestrzeni sygnałów wejściowych. Przekroje te są dwuwymiarowe, granice decyzyjne zaś powstają w trójwymiarowej przestrzeni sygnałów wejściowych. Neurony warstwy ukrytej wyznaczają w niej płaszczyzny decyzyjne, a efektem ich kombinacji w neuronie



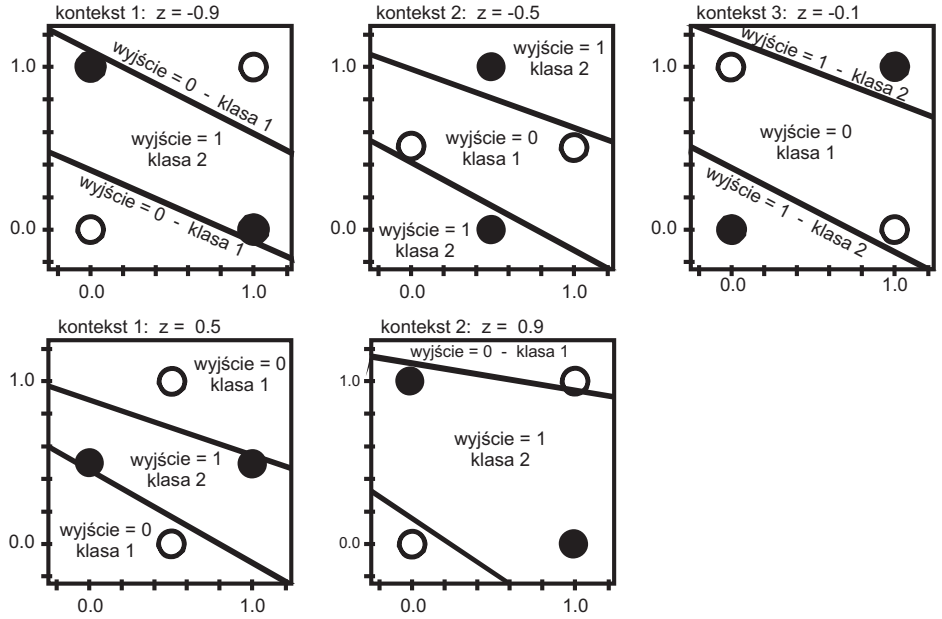
Rysunek 6.14. Przykład XOR w dwóch kontekstach z kontekstem przejściowym – granice decyzyjne sieci kontekstowej o strukturze 2-2-1:2 (18 współczynników) w poszczególnych kontekstach



Rysunek 6.15. Przykład XOR w dwóch kontekstach z kontekstem przejściowym – granice decyzyjne sieci tradycyjnej o strukturze 3-3-1 (16 wag) w poszczególnych kontekstach

wyjściowym są nieliniowe granice dopasowane do 12 zadanych punktów. Granice te generowane są przez 16 stałych wag neuronu tradycyjnego, z których trzy (po jednej dla każdego neuronu warstwy pierwszej) prowadzą do wejścia kontekstowego.

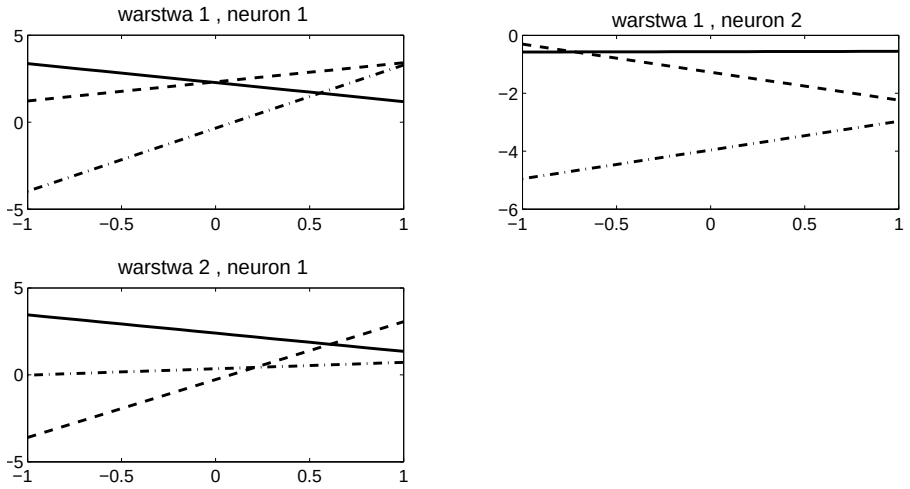
Zauważmy, że dodawanie po jednej funkcji bazowej dla każdego wejścia kontekstowego dość szybko zwiększa liczbę parametrów sieci kontekstowych. Sieć kontekstowa rozwiązująca problem XOR w dwóch kontekstach z rysunku 6.12 miała już 18 współczynników. Wystarczyło jej to również do rozwiązania nieco bardziej skomplikowanego zadania z łagodnym przejściem granic decyzyjnych pomiędzy kontekstami. Dodawanie funkcji bazowych nie zwiększa przestrzeni sygnałów wejściowych, lecz przestrzeń parametrów.



Rysunek 6.16. Przykład XOR w trzech kontekstach z kontekstem przejściowym – granice decyzyjne sieci kontekstowej o strukturze 2-2-1:3 (27 współczynników) w poszczególnych kontekstach

W przestrzeni sygnałów wejściowych, obejmującej wejścia podstawowe oraz kontekstowe, sieć tradycyjna modeluje jednocześnie zarówno granice decyzyjne pomiędzy wejściami podstawowymi, jak i zmianę tych granic wraz z kontekstem. W sieci kontekstowej zadania te są rozdzielone pomiędzy tradycyjną strukturę sieci (wagi), a strukturę kontekstową (funkcje bazowe i współczynniki). Struktura tradycyjna modeluje granice decyzyjne pomiędzy wejściami podstawowymi, natomiast struktura kontekstowa – zmianę położenia tych granic pomiędzy kontekstami.

Rozmiar sieci kontekstowej jest większy od rozmiaru sieci tradycyjnej korzystającej z tej samej informacji kontekstowej. Ponadto szybciej wzrasta on wraz z dodawaniem kolejnych wejść kontekstowych niż rozmiar sieci tradycyjnej. Jednak dzięki rozdzieleniu modelowania obu wspomnianych zależności między wejściami a wyjściami, granice decyzyjne sieci kontekstowej nie są niepotrzebnie komplikowane w pojedynczych kontekstach, a jednocześnie zmieniają się wraz ze zmianą kontekstu. Sieć nie wykazuje oznak przeuczenia (ang. *overfitting*), czyli nadmiernego dopasowania do danych, którego wystąpienie w sieciach tradycyjnych sugerują wykresy ich granic decyzyjnych (rys. 6.15 i 6.19).

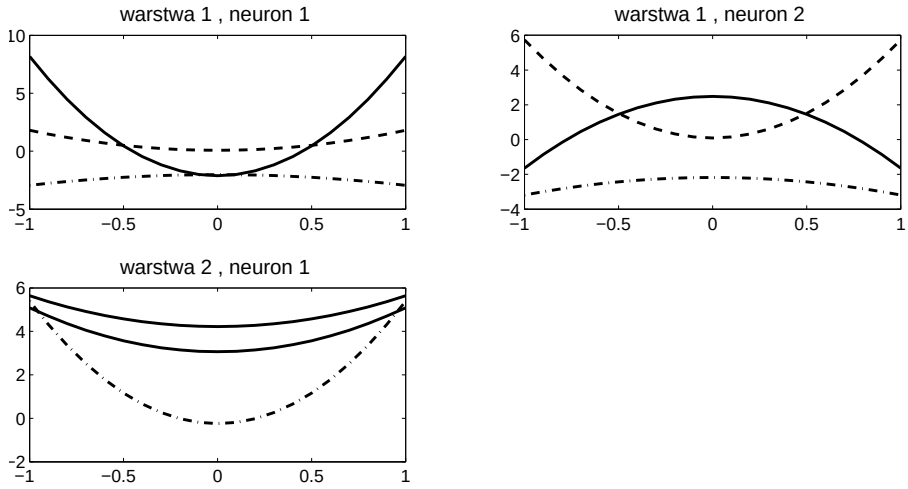


Rysunek 6.17. Przykład XOR w dwóch kontekstach z kontekstem przejściowym – zależność wag poszczególnych neuronów sieci o strukturze 2-2-1:2 (9 wag, 18 współczynników) od wartości wejścia kontekstowego

W sieci tradycyjnej po ustaleniu granic decyzyjnych w jednym kontekście i dodaniu wejścia kontekstowego, w innych kontekstach byliśmy w stanie poprowadzić granice decyzyjne jedynie równoległe do tej pierwszej. W każdym kolejnym kontekście (przekroju) mogliśmy zaś rozdzielić na wszystkie sposoby tylko jeden dodatkowy punkt. Aby móc generować bardziej złożone granice decyzyjne, trzeba w sieciach tradycyjnych dodawać więcej neuronów do warstwy ukrytej, przy czym ich liczba rośnie szybciej niż złożoność problemu.

W sieci kontekstowej o ustalonej liczbie wejść podstawowych i kontekstowych, projektant sieci może dodatkowo zwiększać jej zdolności rozdzielcze, operując wektorem funkcji bazowych. Na przykład w sieci o dwóch wejściach podstawowych i jednym kontekstowym rozszerzenie wektora funkcji bazowych o kolejną funkcję dodaje kolejny kontekst (przekrój), w którym sieć jest w stanie dokonać wszystkich możliwych dychotomii kolejnych trzech punktów na płaszczyźnie. Innymi słowy, sieć może w kolejnym kontekście zamodelować granice decyzyjne o parametrach niezależnych od granic w pozostałych kontekstach. Jeśli wektor funkcji bazowych sieci rozwiązującej zadanie XOR w dwóch kontekstach (rys. 6.12) rozszerzymy o kolejną funkcję bazową (z zestawu funkcji wielomianowych)

$$\mathbf{v}(\mathbf{z}) = \begin{bmatrix} 1 \\ z \end{bmatrix} \quad \rightarrow \quad \mathbf{v}(\mathbf{z}) = \begin{bmatrix} 1 \\ z \\ z^2 \end{bmatrix},$$



Rysunek 6.18. Przykład XOR w trzech kontekstach z dwoma kontekstami przejściowymi – zależność wag poszczególnych neuronów sieci o strukturze 2-2-1:2 (9 wag, 18 współczynników) od wartości wejścia kontekstowego

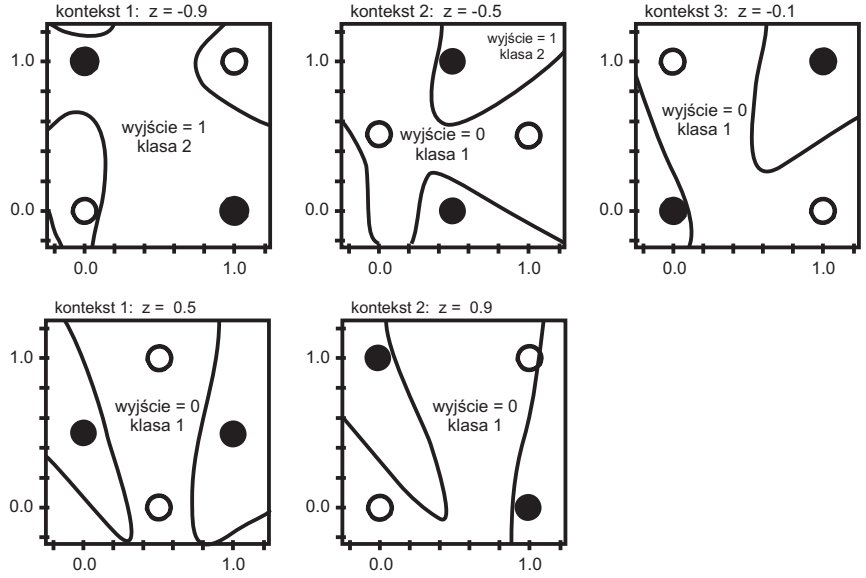
to sieć bez problemu rozwiąże problem XOR w trzech kontekstach, nawet w trudniejszej wersji – z łagodnym przejściem między kontekstami. Strukturę sieci będzie teraz można zapisać jako 2-2-1:3, jej rozmiar zaś wzrośnie do 27 współczynników.

Nie zawsze jednak konieczne jest zwiększanie rozmiaru sieci przez dodawanie kolejnych funkcji bazowych. We wspomnianym przed chwilą przykładzie, zamiast dodawać kolejną funkcję bazową, wystarczy jedną z nich zamienić na funkcję kwadratową,

$$\mathbf{v}\left(\mathbf{z}\right) = \begin{bmatrix} 1 \\ z^2 \end{bmatrix},$$

aby sieć o dwóch funkcjach bazowych, a więc strukturze 2-2-1:2 i rozmiarze 18 współczynników, potrafiła rozwiązać to samo zadanie. Dzięki zastosowaniu kwadratowej funkcji bazowej sieć może bowiem zamodelować zależność wag od kontekstu pokazaną na rysunku 6.18.

Dla porównania, zadanie to rozwiązuje również sieć tradycyjna o czterech neuronach w warstwie ukrytej, a więc o strukturze 3-4-1, czyli 21 wagach. Jej granice decyzyjne, przedstawione na rysunku 6.19, nie odpowiadają jednak naturze rozwiązywanego zadania – problemowi XOR o granicach przesuwanych w poszczególnych kontekstach, natomiast kształt tych granic zwiastuje początki zjawiska przeuczenia sieci, czyli wymuszonego dopasowania do danych uczących. Widząc



Rysunek 6.19. Przykład XOR w trzech kontekstach z kontekstem przejściowym - granice decyzyjne sieci tradycyjnej o strukturze 3-4-1 (21 wag) w poszczególnych kontekstach

tak wykrzywione granice decyzyjne, moglibyśmy powiedzieć, używając potocznego języka, że sieć tradycyjna „męczy się" przy rozwiązywaniu tego zadania.

Przechodząc w naszych rozważaniach od pojedynczego neuronu do sieci wielowarstwowej tradycyjnej, zauważmy, że dodanie wejść kontekstowych dostarcza bezpośrednio informacji kontekstowej jedynie neuronom warstwy wejściowej. Zwiększa również wymiar przestrzeni sygnałów wejściowych oraz przestrzeni parametrów, lecz jedynie w warstwie wejściowej sieci. W dalszych warstwach dane kontekstowe są już „wymieszane" z danymi z pozostałych wejść.

W sieciach kontekstowych wejścia kontekstowe wpływają (funkcyjnie) na wagi we wszystkich warstwach sieci. Sposób przetwarzania sygnałów z wejść podstawowych jest we wszystkich warstwach sieci jasno odróżniony od sposobu przetwarzania danych z wejść podstawowych.

6.6. Uczenie sieci kontekstowych

W punkcie 6.4 pokazano, że odpowiednia konstrukcja macierzy współczynników dla warstwy sieci oraz użycie iloczynu Kroneckera w modelu sieci kontekstowych

wej umożliwia uproszczenie obliczania wyjścia sieci. Wyjście jednej warstwy sieci tradycyjnej oraz kontekstowej można zapisać wzorami, odpowiednio

$$\hat{\mathbf{y}}_{\text{trad}}(\mathbf{x}) = f\left(\mathbf{w}^T\mathbf{x}\right), \tag{6.13}$$

$$\hat{\mathbf{y}}_{\text{kont}}(\mathbf{x},\mathbf{z}) = f\left(\mathbf{a}^T\left[\mathbf{x}\otimes\mathbf{v}(\mathbf{z})\right]\right), \tag{6.14}$$

gdzie $\mathbf{x}$ jest wektorem wejść podstawowych obu sieci, zawierającym wejście progowe, natomiast $\mathbf{v}(\mathbf{z})$ jest wektorem funkcji bazowych, obliczonym dla wektora wejść kontekstowych $\mathbf{z}$.

W tym miejscu pokażemy, że wspomniane dwa założenia modelu sieci kontekstowych sprawiają, że uczenie sieci kontekstowych jest niewiele bardziej złożone niż uczenie sieci tradycyjnych, zaś sam zapis algorytmów uczenia charakteryzuje się prostotą i przejrzystością, jest także bezpośrednim uogólnieniem w stosunku do modelu tradycyjnego. Podobnie, jak w podanych równaniach, wiele wzorów dla tradycyjnych sieci neuronowych można przekształcić na ich odpowiedniki dla sieci kontekstowych przez proste zastąpienie wektora wejść $\mathbf{x}$ iloczynem Kroneckera $\mathbf{x}\otimes\mathbf{v}(\mathbf{z})$. Za przykład posłuży najczęściej stosowana metoda uczenia sieci tradycyjnych, algorytm najszybszego spadku.

Najpopularniejszą stosowaną podczas uczenia miarą jakości dla sieci tradycyjnych jest błąd średniokwadratowy. Jest on funkcją celu uczenia sieci, minimalizowaną podczas tego procesu. Błąd ten dla sieci jednowarstwowej o M neuronach dla jednej epoki uczenia, w której podano na wejścia sieci jeden przykład ze zbioru uczącego, dany jest wzorem

$$E = \frac{1}{2}\sum_{i=1}^{M} d_i^2 = \frac{1}{2}\sum_{i=1}^{M}\left[y_i - \hat{y}_i(\mathbf{x})\right]^2, \tag{6.15}$$

gdzie y_i jest sygnałem żądanym na wyjściu i-tego neuronu dla danego przykładu, M jest liczbą neuronów wyjściowych sieci, a $\mathbf{x}$ jest wektorem wejściowym dla danego przykładu.

Pochodna funkcji celu E względem wagi $w_{i,j}$ dana jest wzorem

$$\frac{\partial E}{\partial w_{i,j}} = -d_i f'(u_i)\, x_j. \tag{6.16}$$

Jeśli iloczyn dwóch pierwszych składników mnożenia (6.16), czyli błędu na wyjściu i-tego neuronu oraz pochodnej funkcji aktywacji tego neuronu, oznaczymy przez c_i i zbierzemy takie iloczyny dla poszczególnych neuronów warstwy sieci w wektorze $\mathbf{c}$, to wektor gradientu funkcji błędu względem wszystkich wag danej warstwy można będzie zapisać wzorem

$$\nabla E = -\mathbf{c}\mathbf{x}^T. \tag{6.17}$$

Błąd średniokwadratowy dla warstwy neuronów kontekstowych dany jest takim samym wzorem jak dla sieci tradycyjnej, czyli (6.15). Inna jest jedynie zależność wyjścia $\hat{y}_i(\mathbf{x})$ każdego z neuronów od wektorów wejść podstawowych $\mathbf{x}$ oraz kontekstowych $\mathbf{z}$, natomiast gradient funkcji błędu należy obliczać nie względem wag w, lecz względem współczynników a. Jak można pokazać, pochodna funkcji błędu względem współczynnika $a_{i,j,k}$, przybliżającego wagę $w_{i,j}$, równa jest iloczynowi pochodnej względem tej wagi oraz wartości k-tej funkcji bazowej

$$\frac{\partial E}{\partial a_{i,j,k}} = -d_i f'(u_i)\, x_j v_k(\mathbf{z}). \tag{6.18}$$

Jak było pokazane w [5], dzięki zaproponowanej konstrukcji macierzy współczynników warstwy a, gradient funkcji błędu E względem macierzy współczynników a dla całej warstwy sieci można zapisać wzorem bardzo zbliżonym do wzoru dla sieci tradycyjnej (na gradient funkcji błędu względem macierzy wag), czyli

$$\nabla E = -\mathbf{c}\left(\mathbf{x} \otimes \mathbf{v}(\mathbf{z})\right)^T. \tag{6.19}$$

Zależność ta po raz kolejny pokazuje, że model sieci kontekstowej jest uogólnieniem modelu tradycyjnego, gdyż w przypadku użycia wektora funkcji bazowych, składającego się z funkcji stałej, wzór (6.19) w naturalny sposób redukuje się do postaci (6.17).

6.7. Złożoność obliczeniowa sieci kontekstowych

W dotychczasowych rozważaniach pokazano, że model kontekstowych sieci neuronowych jest lepiej dopasowany do struktury rozwiązywanych zadań niż model sieci tradycyjnych. Możemy dzięki temu używać sieci o mniejszej liczbie parametrów, przez co działanie oraz uczenie sieci będą bardziej wydajne.

Dokładną analizę złożoności obliczeniowej samych algorytmów obliczania odpowiedzi oraz uczenia sieci przedstawiono w [5]. W tym miejscu przypomnijmy jedynie, że oszacowanie liczby operacji mnożenia wykonywanych podczas używania sieci (obliczania odpowiedzi sieci nauczonej po podaniu sygnału na wejście) jest o 20% większe dla średnich sieci kontekstowych w porównaniu do sieci tradycyjnych o tym samym rozmiarze. Różnica ta zanika wraz ze zwiększaniem rozmiaru sieci. Podczas używania sieci kontekstowych wydajniejszy jest jednoetapowy algorytm obliczania wyjść sieci – wykorzystujący iloczyn Kroneckera wejść podstawowych z wektorem funkcji bazowych i pomijający obliczanie wartości wag w danym kontekście. Podczas uczenia sieci wielowarstwowych metodą propagacji wstecznej błędu nie można już tego etapu pominąć, gdyż do obliczenia domniemanych błędów neuronów warstw ukrytych potrzebna jest znajomość

wartości wag następnej warstwy sieci w bieżącym kontekście. Jednak już samo obliczenie gradientu charakteryzuje się taką samą złożonością obliczeniową jak dla sieci tradycyjnych o tym samym rozmiarze. W przypadku natomiast dużych sieci kontekstowych o wielu funkcjach bazowych przewaga w szybkości działania nad sieciami tradycyjnymi o porównywalnych zdolnościach powiększa się.

Prawdziwą przewagę wykazują sieci kontekstowe z wykorzystaniem algorytmów uczenia opartych na hesjanie. Przedstawiona wcześniej konstrukcja macierzy współczynników A oraz wykorzystanie iloczynu Kroneckera umożliwiają zastąpienie obliczania odwrotności jednej dużej macierzy obliczaniem odwrotności dwóch mniejszych. Ze względu na złożoność obliczeniową samego procesu odwracania macierzy pozwala to na kilkunasto- lub nawet kilkudziesięciokrotne przyspieszenie jednej epoki uczenia sieci, w zależności od złożoności jej struktury.

Zbliżona do sieci tradycyjnych bądź lepsza od nich wydajność uczenia sieci kontekstowych, lepsze dopasowanie modelu sieci do rozwiązywanych zadań, efektywna nauka wynikająca z szybszej zbieżności do żądanego rozwiązania, pokazują przydatność sieci kontekstowych podczas przetwarzania intensywnych strumieni danych.

6.8. Przykład zastosowania sieci kontekstowych do wyceny opcji

W niniejszym podrozdziale przedstawiony zostanie praktyczny przykład zastosowania sieci kontekstowych do modelowania silnie nieliniowych zależności finansowych. Sieć zostanie zastosowana do wyceny opcji walutowych na giełdzie *forex*. Jest to rynek nieregulowany o dużej płynności, na którym kwotowania par walutowych oraz innych instrumentów odbywają się w trybie ciągłym, przy czym notowania te zmieniają się kilka razy na sekundę. Przetwarzane informacje można więc nazwać intensywnym strumieniem danych. Operacje wykonywane na tych danych mogą obejmować zarówno używanie sieci neuronowych nauczonych konkretnych zadań (wyceny opcji, bądź np. szacowania zmienności rynku), jak i ciągłe douczanie sieci w celu dostosowania ich do zmieniających się warunków rynkowych.

Opcja jest pochodnym instrumentem finansowym – jest to umowa dająca nabywcy opcji prawo, lecz nie obowiązek, do kupna (opcja *call*) lub sprzedaży (opcja *put*) od wystawcy opcji pewnego aktywu (akcji, waluty, indeksu, towaru lub innego instrumentu finansowego), zwanego instrumentem bazowym, po określonej cenie w określonym terminie w przyszłości. Cena ta nazywana jest *ceną wykonania* i może być różna od bieżącej rynkowej ceny tego aktywu. Opcja jest rodzajem zakładu dotyczącego ceny danego instrumentu w przyszłości, przy czym ryzyko nie jest tu równo rozłożone pomiędzy nabywcę a wystawcę opcji.

Gdy nabywca spodziewa się dużej zmienności ceny instrumentu bazowego, kupuje opcję *call*, licząc na wzrost ceny do poziomu przekraczającego tzw. *breakeven point*, będącego sumą ceny instrumentu bazowego oraz samej opcji w momencie jej kupna. Jeśli inwestor spodziewa się analogicznego spadku ceny instrumentu bazowego, nabywa opcję *put*. Wystawca opcji na początku tego „zakładu" otrzymuje od nabywcy premię opcyjną – równą cenie opcji w momencie jej wystawienia – i spodziewa się niewielkiej zmienności ceny danego instrumentu. Cena aktywu w momencie wykonania opcji decyduje o zysku lub stracie zarówno nabywcy, jak i wystawcy. Jeśli cena podąża w kierunku założonym przez nabywcę, skorzysta on z prawa do realizacji opcji, zarabiając w dniu jej wykonania różnicę między ceną instrumentu bazowego w tym dniu a ceną w dniu zakupu opcji pomniejszoną o cenę samej opcji. Zysk nabywcy może być teoretycznie nieskończony, natomiast jego strata ograniczona do ceny opcji, którą zapłacił na samym początku, a która przepadnie, jeśli cena instrumentu podąży w niekorzystnym dla niego kierunku – wtedy w dniu wykonania opcji nie skorzysta on po prostu z przysługującego mu prawa do zakupu bądź sprzedaży instrumentu bazowego. Wyboru takiego nie ma wystawca opcji – on musi kupić lub sprzedać instrument bazowy w umówionym dniu po omówionej wcześniej cenie, jeśli tylko nabywca zechce skorzystać z prawa do realizacji opcji. Wystawcy zależy więc na tym, aby cena instrumentu bazowego w dniu wykonania opcji była jak najbardziej zbliżona do ceny w dniu zakupu opcji – oznacza to bowiem dla niego zysk równy cenie opcji w dniu jej zakupu i jest to maksymalny zysk, jaki może on osiągnąć. Każdy większy ruch tej ceny w kierunku przeciwnym do założonego przez wystawcę powiększa jego stratę, przy czym teoretycznie może ona być nieograniczona.

Na bazie przedstawionej podstawowej definicji buduje się różne strategie opcyjne. Na obecnym poziomie rozwoju rynku wszystkie operacje zakupu, wystawienia oraz realizacji opcji odbywają się wirtualnie – nabywca opcji nie ma kontaktu z wystawcą, łączy ich broker i to on dokonuje wyceny opcji oraz rozliczenia całej transakcji, jej uczestnicy zaś najczęściej w ogóle nie widzą instrumentu bazowego, o który grają (np. uncji złota). Wycena opcji ma w tym procesie kluczowe znaczenie, przy czym jest to skomplikowany matematycznie proces, na który wpływ ma wiele czynników.

Najczęściej (w rozważaniach teoretycznych) jest do tego celu wykorzystywany model wyceny opcji Blacka–Scholesa–Mertona, zaprezentowany w 1973 roku w pracy [3]. Za teorię modelowania rynku dwaj z trójki jej twórców otrzymali w 1997 roku nagrodę Nobla (byli to R. Merton i M. Scholes; F. Black już wtedy nie żył). Słynny wzór Blacka–Scholesa umożliwia sprawiedliwą wycenę opcji typu europejskiego. Za sprawiedliwą uważana jest cena nieumożliwiająca arbitrażu, czyli taka, po której kupno lub sprzedaż nie powinny przynieść zysków ani strat.

Cena opcji kupna (*call*) dana jest wzorem

$$c = SN(d_1) - Xe^{-rT}N(d_2) \ , \tag{6.20}$$

natomiast cena opcji sprzedaży (*put*) wzorem

$$p = Xe^{-rT}N(-d_2) - SN(-d_1) \ , \tag{6.21}$$

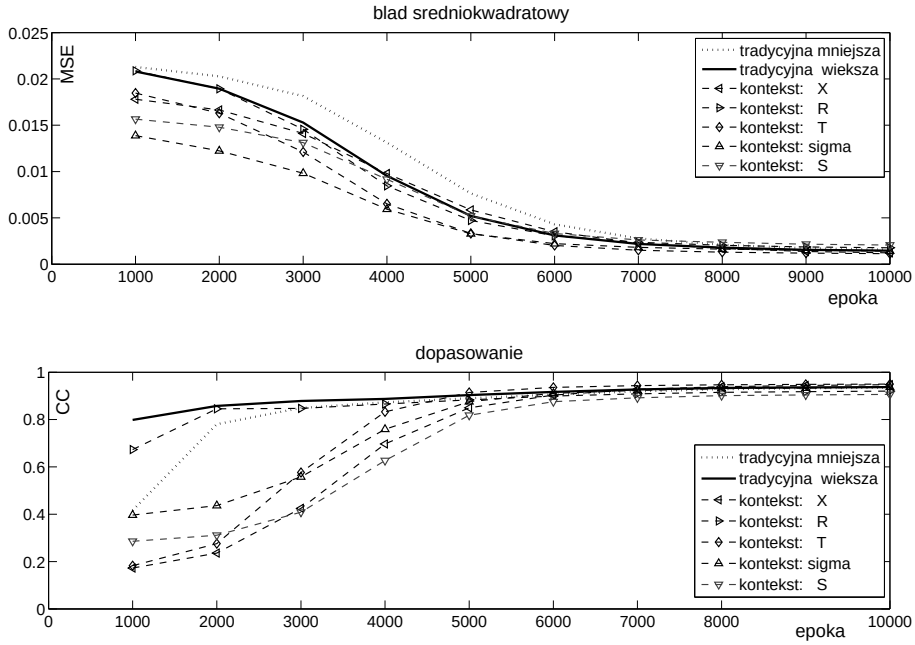
gdzie

$$\begin{aligned} d_1 &= \frac{\ln\left(\frac{S}{X}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \ , \\ d_2 &= \frac{\ln\left(\frac{S}{X}\right) + \left(r - \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} = \\ &= d_1 - \sigma\sqrt{T}, \end{aligned}$$

przy czym: S to obecna cena instrumentu bazowego, X – cena wykonania, r – stopa procentowa wolna od ryzyka (stopa procentowa, na którą wszyscy uczestnicy rynku mogą pożyczać i inwestować bez ryzyka, np. w papiery skarbowe), T – czas pozostający do wygaśnięcia opcji, natomiast σ to procentowy współczynnik zmienności przyszłych cen (odchylenie standardowe). $N(\cdot)$ oznacza dystrybuantę standardowego rozkładu normalnego.

Największe kłopoty podczas wyznaczania cen opcji według podanych wzorów sprawia predykcja przyszłej zmienności instrumentu bazowego. Zmienność ma kluczowy wpływ na wycenę opcji i to właśnie jej znaczenie prowadzi do potocznego stwierdzenia, że na rynku instrumentów pochodnych handluje się nie ceną, lecz zmiennością. Zmienność historyczna (oparta na przeszłych notowaniach) nie przynosi dobrych efektów w prognozowaniu przyszłej zmienności instrumentu, gdyż zakłada niezmienność parametrów rynku w przyszłości. Popularną metodą szacowania przyszłej zmienności stało się rozwiązywanie równania Blacka–Scholesa „w drugą stronę" – uznając obserwowaną cenę rynkową opcji za sprawiedliwą oraz zakładając znajomość pozostałych parametrów, ze wzoru Blacka–Scholesa można wyznaczyć przyszłą zmienność ceny instrumentu bazowego. Tak uzyskaną zmienność nazywa się *zmiennością implikowaną*.

Przedstawiony model wyceny opcji oparty jest na kilku założeniach, które sprawiają, że pomimo uznania w środowisku akademickim oraz popularności w wielu teoriach rynkowych, w wielu sytuacjach nie sprawdza się on i jest stale podważany przez praktyków. Założenia te przewidują m. in., że: ceny akcji podlegają błądzeniu przypadkowemu (model Blacka–Scholesa–Mertona bazuje na teorii

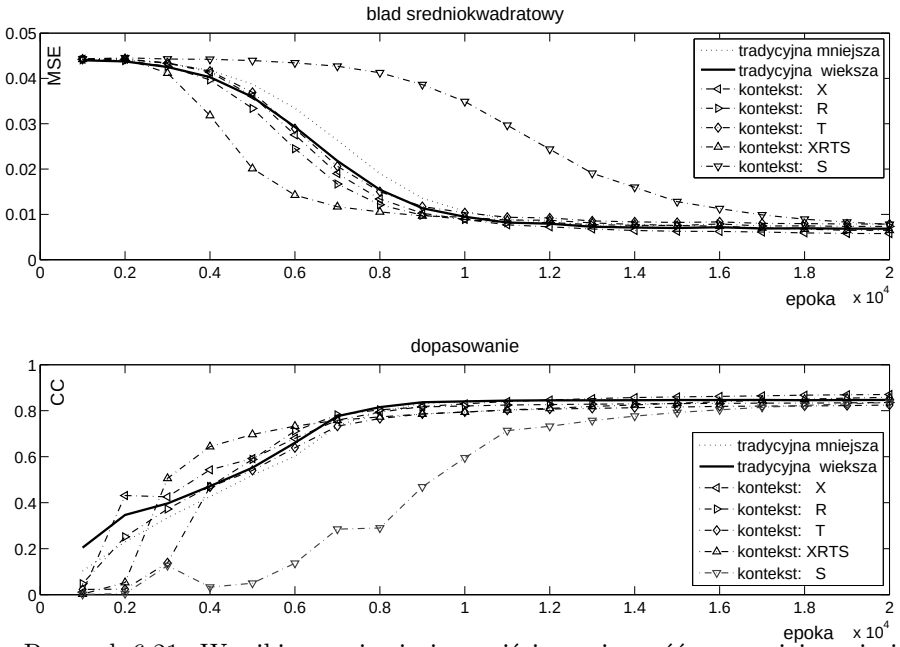


Rysunek 6.20. Wyniki uczenia sieci – wyjście: cena opcji call

ruchów Browna), przez co dla krótkiego okresu mają charakter rozkładu normalnego, natomiast dla dowolnego przyszłego momentu charakteryzują się rozkładem logarytmiczno-normalnym; rynek jest doskonale płynny; zarówno zmienność, jak i stopa wolna od ryzyka nie zmieniają się. Sami autorzy przyznają, że większość z tych założeń jest nierealna. Najczęściej podważany jest logarytmiczno–normalny rozkład zmian ceny oraz stałość współczynnika zmienności w czasie.

Tradycyjne sieci neuronowe były i są używane w wielu zagadnieniach związanych z wyceną opcji na bazie modelu Blacka–Scholesa, a przede wszystkim do konstruowania lepszych od niego modeli wyceny opcji. Większość badań prowadzi do próby nauczenia sieci neuronowych rynkowych wycen opcji. Najczęściej stosuje się różnego rodzaju modele hybrydowe, używające sieci neuronowych do predykcji różnic między modelem teoretycznym a cenami opcji obserwowanymi na rynku dla różnych rodzajów instrumentów ([7], [9]). Jak już wspomniano, stosuje się tradycyjne sieci neuronowe również do „wstecznego" określania zmienności implikowanej ze wzoru Blacka–Scholesa na podstawie rynkowej ceny opcji oraz pozostałych parametrów ([1]). Obszerne zestawienie literatury dotyczącej podobnych zastosowań można znaleźć na przykład w [2] lub [8].

W celu zilustrowania przydatności sieci kontekstowych do modelowania finansowych strumieni danych przedstawione zostaną wyniki uczenia sieci wyceny opcji

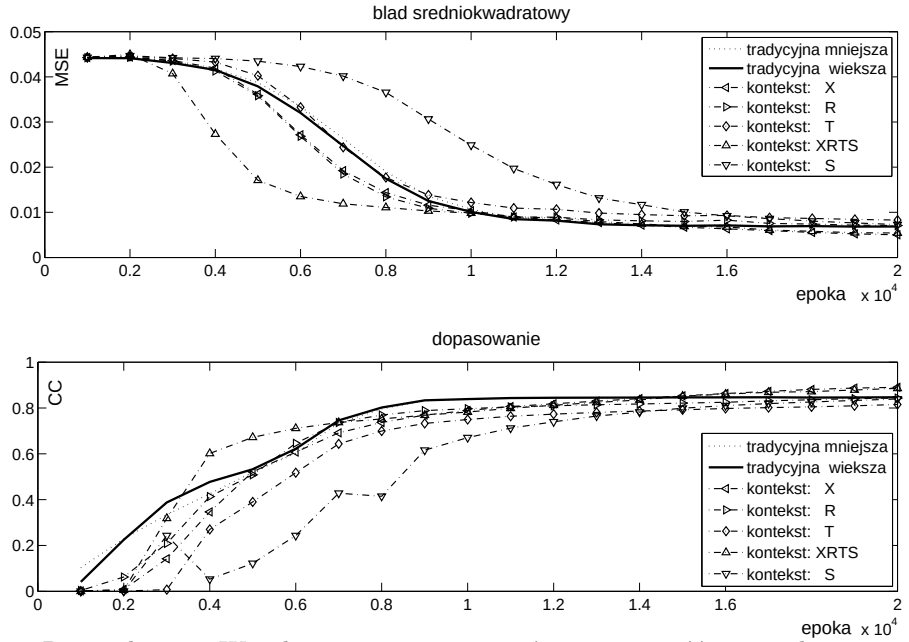


Rysunek 6.21. Wyniki uczenia sieci – wyjście: zmienność σ – mniejsze sieci

call na podstawie wzoru Blacka–Scholesa dla najpopularniejszego instrumentu na giełdzie walutowej forex, czyli EURUSD. Porównane zostały sieci tradycyjne oraz kontekstowe o zbliżonej architekturze oraz rozmiarze. Obu grupom sieci postawione zostały dwa zadania: 1) wyznaczenie ceny opcji kupna przy znajomości jej parametrów oraz 2) wyznaczenie zmienności implikowanej na podstawie ceny opcji kupna oraz pozostałych parametrów. Są to zadania czysto teoretyczne – sieci uczyły się zależności wynikających wprost ze wzorów – i traktujemy je jako badania wstępne przed próbą modelowania rzeczywistych zależności obserwowanych na rynku.

W pierwszym zadaniu na wejścia sieci podane zostały następujące parametry: X – cena wykonania opcji, R – stopa procentowa wolna od ryzyka, T – czas pozostały do wykonania opcji, σ oraz cena bieżąca instrumentu S. Dane uczące oraz testujące dla sieci zostały ustalone w granicach zbliżonych do poziomów cen obserwowanych na przełomie lat 2008 i 2009, czyli: ceny X oraz S – w zakresie od 1,0000 do 1,3000 [1], natomiast pozostałe parametry: stopa R – od 0,5% do 5,0% w skali roku, czas T – od 7 do 90 dni, zmienność σ – od 10% do 200%.

[1] Zapis ten jest charakterystyczny dla kwotowań walut, zwykle notowanych do 4 miejsc po przecinku. W przypadku pary walutowej EURUSD kwotowanie 1,3000 oznacza, że za jednego dolara amerykańskiego trzeba zapłacić 1,3 euro.



Rysunek 6.22. Wyniki uczenia sieci – wyjście: zmienność σ – większe sieci

Porównane zostały sieci tradycyjne oraz kontekstowe o 5 wejściach i 5 neuronach w warstwie ukrytej, przy czym w sieciach kontekstowych każda wielkość została po kolei podana na wejście kontekstowe (badane były jedynie struktury sieci z jednym wejściem kontekstowym i pozostałymi traktowanymi jako wejścia podstawowe). W sieciach tradycyjnych wszystkie wielkości były oczywiście podawane na wejścia podstawowe. Sieci kontekstowe wykorzystywały dwie funkcje bazowe: funkcję stałą oraz liniową wejścia kontekstowego. Wektor funkcji bazowych dany więc był jako

$$\mathbf{v}\left(\mathbf{z}\right) = \left[\begin{array}{c} 1 \\ z \end{array} \right].$$

Wszystkie sieci korzystały z sigmoidalnej funkcji aktywacji. Dla porównania zdolności sieci tradycyjnych i kontekstowych o tej samej liczbie parametrów do analizy dołączona została również sieć tradycyjna o 9 neuronach w warstwie ukrytej. Liczba parametrów (a więc i wymiar VC-dim) takiej sieci tradycyjnej zbliżona jest do sieci kontekstowej o 5 neuronach ukrytych, zbliżone są więc tym samym potencjalne możliwości obu sieci.

Struktury analizowanych sieci oraz wyniki ich uczenia podane są w tabeli 6.1. Oznaczenie 4:1-5-1 (2) odnosi się do sieci kontekstowej o czterech wejściach tradycyjnych, jednym kontekstowym, pięciu neuronach w warstwie ukrytej i jednym

Tabela 6.1. Wyniki uczenia sieci – wyjście: cena opcji call

Sieć	VC dim	Wejścia podst.-kont.	Błąd MSE	Wsp. dop. CC	Liczba epok
trad. 5-5-1	36	XRTσS	0,001129	0,948988	100 000
trad. 5-9-1	64	XRTσS	0,001038	0,953048	100 000
kont. 4:1-5-1 (2)	62	RTσS – X	0,000985	0,955357	15 000
kont. 4:1-5-1 (2)	62	XTσS – R	0,001081	0,951059	100 000
kont. 4:1-5-1 (2)	62	XRσS – T	0,001008	0,954295	15 000
kont. 4:1-5-1 (2)	62	XRTS – σ	0,001026	0,954348	19 000
kont. 4:1-5-1 (2)	62	XRTσ – S	0,001017	0,954057	64 000

wyjściowym, w której zależność wag od kontekstu modelowana jest za pomocą dwóch funkcji bazowych. W kolumnie *wejścia* określone jest, które zmienne zostały użyte jako wejścia podstawowe dla sieci, a które – jako kontekstowe.

W celu zachowania możliwie jednolitych warunków testowania obu rodzajów sieci, wszystkie one uczone były tą samą metodą – wstecznej propagacji błędu z momentum z jednakowymi parametrami oraz kolejnością pokazywania przykładów uczących (po 5 przykładów w jednej epoce). Dane uczące podzielone zostały na trzy części w następujących proporcjach: 60% na dane uczące, 10% na dane sprawdzające i 30% na dane testujące. Sieci uczone były na przykładach z ciągu uczącego do momentu osiągnięcia błędu średniokwadratowego równego 0,001 na ciągu sprawdzającym, lecz nie dłużej niż przez 100 000 epok. Tabele wyników zawierają rezultaty testowania sieci na ciągach testujących. Za miary jakości służą błąd średniokwadratowy MSE oraz współczynnik dopasowania CC – współczynnik korelacji między odpowiedziami sieci a żądanymi odpowiedziami w ciągu testującym.

Żadna z sieci tradycyjnych nie osiągnęła zakładanego poziomu błędu po 100 000 epok, choć zbliżyły sie one do celu. Aż cztery z pięciu sieci kontekstowych osiągnęło zakładany poziom błędu oraz dopasowania po znacznie krótszym uczeniu (trzy z tych sieci w czasie 5–6 razy krótszym niż sieci tradycyjnych). Świadczy to o dobrym dopasowaniu struktury sieci do modelowanego zjawiska – zależności występujące między ceną opcji a jej parametrami mają charakter kontekstowy. Przebieg uczenia wszystkich sieci pokazany jest na rys. 6.20.

Kolejnym zadaniem dla sieci było modelowanie odwrotnej zależności – wyznaczenie zmienności implikowanej σ ze wzoru Blacka–Scholesa na podstawie ceny

opcji oraz pozostałych parametrów opcji. Użyto w tym celu sieci tradycyjnych oraz kontekstowych o tych samych założeniach, jak w poprzednim zadaniu (z wyjątkiem sieci kontekstowej nr 4). Danymi wejściowymi dla sieci były tym razem: cena opcji call oraz pozostałe parametry: X, R, T, S. Zmienną wyjściową była zmienność σ. Sieci tradycyjne przyjmowały wszystkie wejścia jako podstawowe, natomiast w sieciach kontekstowych po kolei każda zmienna podawana była na wejście kontekstowe, a pozostałe na wejścia podstawowe. Jedynie czwarta sieć kontekstowa przyjmowała na wejście podstawowe cenę opcji call, natomiast pozostałe cztery parametry na wejścia kontekstowe. Wektor funkcji bazowych tej sieci składał się z 5 elementów: funkcji stałej oraz funkcji liniowych poszczególnych wejść. Z racji ograniczenia liczby wejść tradycyjnych do 1, zwiększenie liczby funkcji bazowych nie spowodowało znaczącego wzrostu wymiaru VC-dim dla tej sieci w porównaniu do pozostałych.

Tabela 6.2. Wyniki uczenia sieci – wyjście: zmienność σ – mniejsze sieci

Sieć	VC dim	Wejścia podst.-kont.	Błąd MSE	Wsp.dop. CC	Liczba epok
trad. 5-5-1	36	call XRTS	0,003226	0,928137	100 000
trad. 5-9-1	64	call XRTS	0,002820	0,937342	100 000
kont. 4:1-5-1 (2)	62	call RTS–X	0,002539	0,942799	100 000
kont. 4:1-5-1 (2)	62	call XTS–R	0,002466	0,945271	100 000
kont. 4:1-5-1 (2)	62	call XRS–T	0,003713	0,916543	100 000
kont. 1:4-5-1 (5)	80	call –XRTS	0,002397	0,946211	100 000
kont. 4:1-5-1 (2)	62	call XRT–S	0,003269	0,927084	100 000

Zbadane zostały dwie kontekstowe struktury sieci – oznaczone jako *mniejsze* oraz *większe*. Większe sieci używały wektora funkcji bazowych rozszerzonego o jedną funkcję bazową. Zawierał on trzy funkcje bazowe: funkcję stałą, liniową oraz funkcję kwadratową wejścia kontekstowego. Czwarta sieć kontekstowa używała natomiast wektora funkcji bazowych o 9 składnikach: funkcji stałej, liniowych funkcji wszystkich czterech wejść kontekstowych oraz ich funkcji kwadratowych.

Jak pokazują wyniki uczenia, zawarte w tabelach 6.2 i 6.3 oraz pokazane na rysunkach 6.21 i 6.22, żadna z sieci nie osiągnęła zakładanego poziomu błędu po 100 000 epok, jednak prawie wszystkie sieci kontekstowe uczyły się lepiej niż sieci tradycyjne, natomiast to właśnie czwarta sieć kontekstowa (wykorzystująca

Tabela 6.3. Wyniki uczenia sieci – wyjście: zmienność σ – większe sieci

Sieć	VC dim	Wejścia podst.-kont.	Błąd MSE	Wsp.dop. CC	Liczba epok
trad. 5-5-1	36	call XRTS	0,003226	0,928137	100 000
trad. 5-9-1	92	call XRTS	0,003479	0,922274	100 000
kont. 4:1-5-1 (3)	93	call RTS–X	0,002288	0.948362	100 000
kont. 4:1-5-1 (3)	93	call XTS–R	0,002376	0.947270	100 000
kont. 4:1-5-1 (3)	93	call XRS–T	0,002596	0.942546	100 000
kont. 1:4-5-1 (9)	144	call –XRTS	0,002124	0.952016	100 000
kont. 4:1-5-1 (3)	93	call XRT–S	0,002467	0.944940	100 000

wszystkie zmienne oprócz ceny jako zmienne kontekstowe) osiągnęła najmniejszy błąd i najlepsze dopasowanie w najmniejszej liczbie epok.

Na podstawie przedstawionych wyników badań można stwierdzić, że zarówno w zadaniu wyceny opcji, jak i w odwrotnym zadaniu wyznaczania zmienności implikowanej na podstawie ceny opcji, sieci kontekstowe wykazały swoją przydatność. W krótszym czasie uczenia mniejsze sieci kontekstowe osiągały mniejszy błąd oraz lepsze dopasowanie do modelowanych zależności niż sieci tradycyjne. Świadczy to o tym, że modelowane związki między zmiennymi wejściowymi mają rzeczywiście charakter zależności kontekstowych. Większa wydajność sieci kontekstowych wynika więc z lepszego dopasowania architektury sieci do modelowanych zależności, a także z zastosowanych algorytmów uczenia, dzięki którym uczenie bardziej skomplikowanych strukturalnie sieci charakteryzuje się złożonością obliczeniową zbliżoną do sieci tradycyjnych.

W praktycznym stosowaniu sieci kontekstowych w zadaniach związanych z wyceną opcji, możemy więc liczyć na lepszą wydajność zarówno podczas używania nauczonych już sieci do wyceny opcji (wydajniejsze przetwarzanie napływającego strumienia danych), jak i podczas doraźnego douczania sieci na nowych, stale napływających danych.

6.9. Podsumowanie

W pracy przedstawiono zarys metodologii kontekstowego przetwarzania danych. Zaprezentowano model kontekstowego neuronu oraz kontekstowych sieci wielowarstwowych, przeanalizowano dogłębnie sposób ich działania oraz prze-

twarzania danych. Pokazano algorytmy uczenia, których wydajność jest porównywalna, a czasem lepsza niż dla sieci tradycyjnych o zbliżonych rozmiarach. Możliwości wykorzystania sieci kontekstowych do modelowania złożonych zależności finansowych zilustrowano przykładem wyceny opcji walutowych. Uzyskane wyniki są wstępem do dalszych badań, np. nad wykorzystaniem sieci kontekstowych do wyceny opcji na podstawie zmienności historycznej lub do szacowania zmienności implikowanej na podstawie ceny opcji. Przedstawione rozważania pokazały przydatność sieci kontekstowych do przetwarzania intensywnych strumieni danych.

Bibliografia

[1] Amornwattana S., Enke D., Dagli C. *A hybrid option pricing model using a neural network for estimating volatility* , International Journal of General Systems, Volume 36, Number 5, October 2007, 558–573(16).

[2] Bennell J.A., Sutcliffe C.M., *Black-Scholes Versus Artificial Neural Networks in Pricing FTSE 100 Options*, Intelligent Systems in Accounting, Finance and Management, 12, 243–260, 2004.

[3] Black F., Scholes M., *The pricing of options and corporate liabilities*, Journal of Political Economy, Vol. 81, No. 3, pp. 637–659, March 1973.

[4] Ciskowski P., *Vapnik-Chervonenkis Dimension of a Context-Dependent Perceptron*, Proc. of Third International and Interdisciplinary Conf. on Modeling and Using Context, CONTEXT 2001, Dundee, UK, July 2001.

[5] Ciskowski P., *Learning of Context-Dependent Neural Nets*, rozprawa doktorska, Poltechnika Wrocławska, Wrocław, 2002.

[6] Ciskowski P., Rafajłowicz E., *Context–Dependent Neural Nets – Structures and Learning*, IEEE Transactions on Neural Networks, Vol. 15 (6), 1367–1377, 2004.

[7] Lajbcygier P., Connor J.T., *Improved Option Pricing Using Artificial Neural Networks and Bootstrap Methods*, Int. Journal of Neural Systems, Vol. 8 (4), 1997.

[8] Lajbcygier P., *Improving Option Pricing With the Product Constrained Hybrid Neural Network*, IEEE Transactions on Neural Networks, Vol. 15 (2), March 2004.

[9] Malliaris M., Salchenberger L., *Beating the best: A neural network challenges the Black-Scholes formula*, Proc. of the Ninth Conf. on Artificial Intelligence for Applications, Orlando FL, USA, 1–5 March 1993.

[10] Pao Y.H., *Adaptive Pattern Recognition and Neural Networks*, Addison Wesley, Reading, 1989.

[11] Rafajłowicz E., *Context dependent neural nets – problem statement and examples*, Proc. of the Third Conf. Neural Networks and Their Applications, Zakopane, Poland, May 1999.

[12] Spencer Jr. B., Dyke S., Sain M., Carlson J., *Phenomenological Model of a Magnetorheological Damper*, Journal of Engineering Mechanics, ASCE, Vol. 123 (3), 230–238, 1997.

[13] Watrous R., Towell G., *A Patient-Adaptive Neural Network ECG Patient Monitoring Algorithm*, Computers in Cardiology, Sept. 1995, Vienna, Austria, 10–13.

[14] Yeung D.T., Bekey G.A., *Using a Context-Sensitive Learning for Robot Arm Control*, Proc. of the IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, May 14–19, 1989, 1441–1447.