Raport Badawczy
Research Report

RB/20/2013

Algorytmy detekcji i lokalizacji wycieków w sieciach wodociągowych

J. Studziński, I. Rojek

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences



POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 3810100

fax: (+48) (22) 3810105

Kierownik Zakładu zgłaszający pracę:
Prof. nadzw. dr hab. inż. Jan Studziński

Warszawa 2013

Algorytmy detekcji i lokalizacji wycieków w sieciach wodociągowych

Jan Studziński, Izabela Rojek

Warszawa 2013

Spis treści

1. Wstęp 4
2. Sieci neuronowe 4
2.1. Sieci jednokierunkowe wielowarstwowe typu MLP 5
2.2. Sieć neuronowa samoorganizująca się Kohonena [20] 12
2.3. Podsumowanie sieci neuronowych użytych do eksperymentów 22
3. Eksperymenty dotyczące tworzenia algorytmów lokalizacji awarii w sieci wodociągowej 22
3.1. Stan obecny 22
3.2. Analiza lokalizacji awarii przy wykorzystaniu sieci neuronowej 23
3.2.1. Algorytm lokalizacji awarii sieci wodociągowej 23
3.2.2. Wyznaczenie punktów wrażliwych sieci wodociągowej przy użyciu algorytmu do planowania systemu monitoringu 24
3.2.3. Wykonanie obliczeń hydraulicznych sieci, dla zadanego poziomu obciążenia, za pomocą wybranego modelu hydraulicznego 24
3.2.4. Tworzenie klasyfikatora 28
4. Utworzenie i wybór najlepszego klasyfikatora lokalizującego punkty pomiarowe najbliżej awarii w postaci wybranych sieci neuronowych 34
4.1. Utworzenie i wybór najlepszego klasyfikatora w postaci sieci neuronowych typu MLP 35
4.2. Utworzenie i wybór najlepszego klasyfikatora w postaci sieci neuronowych Kohonena 43
5. Utworzenie i wybór najlepszego klasyfikatora lokalizującego awarie w węźle dla 10 punktów pomiarowych 51
5.1. Utworzenie i wybór najlepszego klasyfikatora w postaci sieci typu MLP 51
5.2. Utworzenie i wybór najlepszego klasyfikatora w postaci sieci Kohonena 59
5.3. Porównanie najlepszych sieci neuronowych MLP i Kohonena dla pliku z danymi oryginalnymi, poprawionymi i poprawionymi bez powtórzeń kolumn 65
6. Utworzenie i wybór najlepszego klasyfikatora lokalizującego awarie w węźle dla 20 punktów pomiarowych 66
6.1. Utworzenie i wybór najlepszego klasyfikatora w postaci sieci typu MLP 66
6.2. Utworzenie i wybór najlepszego klasyfikatora w postaci sieci Kohonena 73
6.3. Analiza wyników sieci neuronowych typu MLP i Kohonena dla pliku z danymi oryginalnymi 76
6.4. Utworzenie i wybór najlepszego klasyfikatora lokalizującego awarie w postaci sieci neuronowych typu MLP dla pliku zmodyfikowanego z usuniętymi 2 kolumnami (usunięta kolumna pp2 i pp7) 78
6.5. Utworzenie i wybór najlepszego klasyfikatora lokalizującego awarie w postaci sieci neuronowych typu Kohonena dla pliku zmodyfikowanego z usuniętymi 2 kolumnami (usunięta kolumna pp2 i pp7) 81

6.6. Analiza wyników sieci neuronowych typu MLP i Kohonena dla pliku z danymi zmodyfikowanymi bez powtórzeń 82
6.7. Porównanie wyników dla danych oryginalnych i zmodyfikowanych dla sieci MLP i Kohonena 83
6.8. Dane oryginalne – sieci MLP i Kohonena 86
7. Podsumowanie wyników dla sieci neuronowych dla 10 i 20 punktów pomiarowych z danymi oryginalnymi i zmodyfikowanymi dla sieci MLP i Kohonena 88
8. Zmodyfikowane sieci Kohonena 91
Literatura 91

1. Wstęp

We współczesnym świecie firmy i ludzie zasypywani są danymi i informacjami z różnych źródeł. Niestety, te cenne dane, których zebranie i zestawienie kosztuje firmy miliony, marnują się w bazach danych i hurtowniach danych. Problemem jest zbyt mała liczba dostępnych i wykwalifikowanych analityków, którzy potrafią przemienić te wszystkie dane w wiedzę. Wnikliwa analiza tych informacji może okazać się pomocna w nawiązaniu relacji z klientem, pozyskaniu jego lojalności i skierowaniu do niego oferty dostosowanej do jego potrzeb [10]. Dlatego właśnie potrzebna, wręcz niezbędna jest eksploracja danych (*ang. data mining*) [3,6]. Takim właśnie przedsiębiorstwem jest przedsiębiorstwo wodociągowe.

Przedsiębiorstwo wodociągowe w zakresie sieci wodociągowej zajmuje się dystrybucją wody dobrej jakości w ilości gwarantującej zaspokojenie potrzeb odbiorców, poprawną eksploatacją sieci wodociągowej zapewniającą właściwe ciśnienie w węzłach odbiorczych, sprawnym usuwaniem awarii oraz planowaniem i wykonywaniem prac związanych z konserwacją, modernizacją i rozbudową sieci. Sterowanie siecią wodociągową jest trudnym i kompleksowym procesem [18].

Bardzo istotnym problemem w zarządzaniu przedsiębiorstwem wodociągowym jest wykrywanie i lokalizacja ukrytych wycieków wody w sieci wodociągowej. Spowodowane tym straty wody w rurociągach, dochodzące niekiedy nawet do 20%, wpływają niekorzystnie na wyniki finansowe przedsiębiorstw wodociągowych, które utraconej wody nie mogą sprzedać. U odbiorców wody straty te odbijają się w zwiększonych opłatach za wyprodukowaną i nie zużytą faktycznie wodę. Szybka lokalizacja i eliminacja ukrytych wycieków wody z nieszczelnych rurociągów przynosi zatem wymierne korzyści ekonomiczne zarówno dostawcom, jak i odbiorcom wody, czyli przedsiębiorstwu wodociągowemu i użytkownikom sieci wodociągowej.

Etapy dotyczące awarii w sieci wodociągowej można określić w następujący sposób:
- wykrycie awarii – stwierdza się stan awaryjny w postaci strat wody, ale nie jest znana jego lokalizacja,
- lokalizacja awarii – stwierdza się stan awaryjny i jego dokładną lub przybliżoną lokalizację za pomocą odpowiednio opracowanych algorytmów, z wykorzystaniem systemów monitoringu, modeli hydraulicznych sieci wodociągowej i w szczególności sieci neuronowych,
- przeciwdziałanie awarii – na podstawie danych historycznych o stanach awaryjnych następuje prognozowanie kolejnych stanów awaryjnych i opracowuje się plany rewitalizacji sieci wodociągowej.

2. Sieci neuronowe

Do opracowania algorytmów lokalizacji awarii w sieci wodociągowej zostały wykorzystane sieci neuronowe. Wybrano jednokierunkowe wielowarstwowe sieci neuronowe ze wsteczną propagacją błędu (MLP) oraz samoorganizujące się sieci Kohonena.

Sieć MLP jest przedstawicielem sieci, w której wykorzystuje się uczenie pod nadzorem (z nauczycielem), a sieć Kohonena reprezentuje uczenie samoorganizujące się.

Uczenie **pod nadzorem** (z nauczycielem)

- W trybie tym oprócz danych wejściowych musimy również wiedzieć, jakie dane chcemy uzyskać na wyjściu sieci.
- Uczenie polega na doborze wag w taki sposób, aby sygnał na wyjściu był najbliższy wartości zadanej.
- Inaczej mówiąc, celem uczenia pod nadzorem jest minimalizacja odpowiednio zdefiniowanej funkcji celu, która umożliwi dopasowanie wartości aktualnych odpowiedzi neuronów wyjściowych do wartości żądanych.

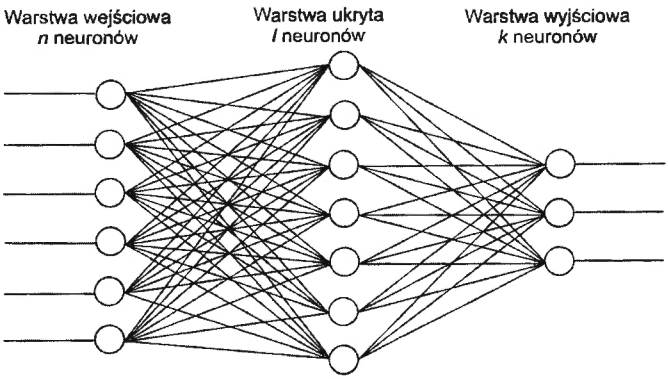
Uczenie **samoorganizujące się**

- W tym przypadku zadane wartości wyjściowe również nie są znane.
- W uczeniu samoorganizującym się tylko jeden neuron może być aktywny, gdy reszta pozostaje w spoczynku.
- W tej metodzie grupa neuronów otrzymuje te same wartości wejściowe. W zależności od wartości wag ich wyjścia różnią się między sobą.
- Wartości wyjść są porównywane ze sobą, zwycięża ten neuron, który na wyjściu ma wartość największą. Neuron zwycięzca przyjmuje wartość 1 a pozostałe 0.

2.1. Sieci jednokierunkowe wielowarstwowe typu MLP

Budowa i działanie sieci

W sieciach jednokierunkowych wielowarstwowych neurony przekazują sygnały tylko między różnymi warstwami. W obrębie tej samej warstwy nie mogą się łączyć ze sobą. Sygnały przekazywane są od warstwy wejściowej do wyjściowej (stąd nazwa jednokierunkowe), przy czym nie występują sprzężenia zwrotne do warstw poprzednich [5,7,8,12,13,19,20]. Typowy schemat sieci jednokierunkowej trójwarstwowej przedstawiono na rys. 1.



Rys. 1. Schemat jednokierunkowej wielowarstwowej sieci neuronowej

Najczęściej wykorzystywanym modelem sieci jest model sieci jednokierunkowej, a dokładnie warstwowej sieci perceptronowej. Ponad 80% zastosowań, to sieci jednokierunkowe wielowarstwowe.

Sieci jednokierunkowe wielowarstwowe ze wsteczną propagacją błędu stanowią naturalne uogólnienie perceptronu Rosenblatta i zwane są również perceptronem wielowarstwowym. Przepływ sygnałów odbywa się od wejścia do wyjścia. Sieci wielowarstwowe zbudowane są z wielu warstw neuronów. Wejście każdego neuronu z danej warstwy ma połączenie z wyjściami wszystkich neuronów w warstwie poprzedniej. W modelu sieci wielowarstwowej pierwsza warstwa jest nazywana warstwą wejściową, ostatnia – wyjściową, a pozostałe – warstwami ukrytymi.

Warstwa wejściowa ma za zadanie tylko wprowadzenie informacji do sieci, nie przetwarza danych, ale podaje je do następnej warstwy.

Warstwa ukryta jest to warstwa posiadająca wagi i podlega adaptacji w procesie uczenia się. W ogólnym przypadku liczba warstw ukrytych może być dowolna, ze względu jednak na skrócenie czasu obliczeń stosuje się przeważnie jedną lub dwie warstwy ukryte. Zarówno liczbę tych warstw, jak i liczbę neuronów w każdej z nich, najczęściej dobiera się eksperymentalnie. Sieć, w której jest zbyt mała liczba neuronów ukrytych, powoduje niezdolność do zgromadzenia dostatecznej wiedzy o rozwiązywanym problemie, a zbyt bogata architektura jest z kolei przyczyną upośledzenia, tzw. generalizacji (sieć zapamiętuje zbyt dokładnie dane treningowe, co powoduje kłopoty z uogólnieniem ich na przypadki nie objęte procesem uczenia). Każdy z neuronów z warstwy wejściowej połączony jest z neuronem z warstwy wyjściowej – taką sieć nazywamy w pełni połączoną.

Warstwa wyjściowa również ma charakter adaptacyjny, ale w przeciwieństwie do warstw ukrytych nie przekazuje ona informacji do kolejnych neuronów, ale wyprowadza je na zewnątrz danej sieci.

Dobór liczby neuronów w warstwie wejściowej jest uwarunkowany wymiarem wektora danych x. Podobnie jest w warstwie wyjściowej, w której liczba neuronów równa

się wymiarowi wektora zadanego d. Problemem pozostaje dobór warstw ukrytych i liczby neuronów w każdej warstwie. W ogólnym przypadku liczba warstw ukrytych może być dowolna, ze względu jednak na skrócenie czasu obliczeń stosuje się przeważnie jedną lub dwie warstwy ukryte. Model neuronu jest typu sigmoidalnego. Składa się on z elementu sumacyjnego, do którego dochodzą sygnały wejściowe x_1, x_2, ..., x_N, tworzące wektor wejściowy $x=[x_1, x_2, ..., x_N]^T$, pomnożone przez przyporządkowane im wagi w_{i1}, w_{i2}, ..., w_{iN}, tworzące wektor wag i-tego neuronu $w_i=[w_{i1}, w_{i2},..., w_{iN}]^T$, oraz przez wartość w_{i0} zwaną progiem. Sygnał wyjściowy sumatora oznaczono u_i (2.1), przy czym

$$u_i = \sum w_{ij} x_j + w_{i0} \qquad (2.1)$$

Sygnał ten jest podawany na blok realizujący nieliniową funkcję aktywacji $f(u_i)$. Funkcja ta przyjmuje postać sigmoidalną (zachowuje wtedy ciągłość i proces uczenia sieci jest łatwiejszy). Postać funkcji stanowi ciągłe przybliżenie funkcji skokowej, przy której sygnał wyjściowy neuronu y_i przyjmuje dowolne wartości z przedziału [0,1] dla funkcji unipolarnej.

Funkcja sigmoidalna unipolarna (2.2), to:

$$f_u(u_i) = \frac{1}{1+\exp(-\beta \cdot u_i)} \qquad (2.2)$$

Metodą uczenia sieci jednokierunkowych sigmoidalnych jest optymalizacja funkcji celu zdefiniowanej dla sieci, minimalizująca błąd między wartościami żądanymi i aktualnie otrzymanymi na wyjściu sieci dla wszystkich danych uczących. Jest to uczenie pod nadzorem. Sygnałom uczącym towarzyszą wartości żądane na wyjściu sieci. Celem uczenia pod nadzorem jest minimalizacja odpowiednio zdefiniowanej funkcji celu, która w wyniku umożliwi dopasowanie wartości aktualnych odpowiednich neuronów wyjściowych (wektor y) do wartości żądanych (wektor d) dla wszystkich p par uczących. Za funkcję celu przyjmuje się błąd średniokwadratowy.

Stąd w praktyce dla uzyskania dobrych zdolności uogólniania sieci należy ograniczać liczbę neuronów ukrytych oraz liczbę powiązań między neuronowych, jak również stosować takie metody wstępnego przetworzenia danych, które umożliwiają zmniejszenie wymiarowości wektora wejściowego sieci. Najlepsze wyniki w uczeniu sieci można osiągnąć przez odpowiedni dobór wstępnych wartości wag sieci. W większości zastosowań używa się głównie losowego doboru wag, przyjmując rozkład równomierny w określonym przedziale liczbowym [0,1] (tzw. randomizacja wag).

Dla sieci zawierających co najmniej jedną warstwę ukrytą metoda propagacji błędu wstecz (Back Propagation – BP) jest jedną z najbardziej popularnych technik uczenia sieci. Może być ona zastosowana do sieci o dowolnej liczbie warstw, przy czym muszą być co najmniej dwie warstwy, w których są ustawiane wartości wag.

Postacie sieci neuronowych:

- programowa (netware) – software, który emuluje neurony i ich połączenia na konwencjonalnych komputerze,

- neurokomputerowa – realizacja sieci neuronowej na poziomie sprzętowym (fizyczne elementy przetwarzające i fizyczne połączenia między nimi).

Sieci wielowarstwowe są uczone przy pomocy różnych algorytmów uczenia z nauczycielem.

Metoda wstecznej propagacji błędów

Do tej pory pokazane zostało uczenie pojedynczych neuronów oraz definiowanie błędu na wyjściu neuronu. W identyczny sposób można zdefiniować błąd dla warstwy ostatniej w przypadku sieci wielowarstwowych. W tym momencie pojawia się problem zdefiniowania błędu dla warstw ukrytych, ponieważ nie wiadomo, jaka powinna być wartość na wyjściu w poszczególnych warstwach ukrytych. W tym przypadku należy zastosować najbardziej rozpowszechnioną technikę uczenia sieci wielowarstwowych neuronowych zwaną metodą wstecznej propagacji błędów. Aby wyprowadzić algorytm metody, należy zdefiniować miarę błędu. Będzie to funkcja Q(w), gdzie w oznacza wektor wszystkich wag sieci wielowarstwowej NN. W trakcie uczenia sieci należy dążyć do znalezienia minimum funkcji Q względem wektora w. Należy rozwinąć funkcję Q w szereg Taylora w najbliższym sąsiedztwie znanego aktualnego rozwiązania w. Rozwinięcie to zostało przedstawione wzdłuż kierunku p w sposób następujący:

$$Q(w+p) = Q(w) + [g(w)]^T p + 0{,}5 p^T H(w) p + ..., \qquad (2.3)$$

gdzie g(w) oznacza wektor gradientu, tzn.

$$g(w) = [\frac{\partial Q}{\partial w_1}, \frac{\partial Q}{\partial w_2}, \frac{\partial Q}{\partial w_3}, ..., \frac{\partial Q}{\partial w_n}]^T, \qquad (2.4)$$

natomiast H(w) jest hesjanem, tzn. macierzą drugich pochodnych

$$H(w) = \begin{bmatrix} \frac{\partial^2 Q}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 Q}{\partial w_1 \partial w_n} \\ \vdots & & \vdots \\ \frac{\partial^2 Q}{\partial w_n \partial w_1} & \cdots & \frac{\partial^2 Q}{\partial w_n \partial w_n} \end{bmatrix}. \qquad (2.5)$$

Modyfikację przeprowadza się w następujący sposób:

$$w(t+1) = w(t) + \eta(t) p(t), \qquad (2.6)$$

gdzie η jest współczynnikiem uczenia.

Modyfikację wag można wykonywać tak długo, aż funkcja Q osiągnie minimum lub jej wartość spadnie poniżej założonego progu. Zadanie sprowadza się więc do wyznaczenia takiego wektora kierunkowego p, aby w kolejnych krokach algorytmu błąd na wyjściu malał. Oznacz to, iż wymagane jest spełnienie nierówności $Q(w(t+1)) < Q(w(t))$ w kolejnych krokach iteracji. Ograniczamy szereg Taylora aproksymujący funkcję błędu Q do rozwinięcia liniowego, tzn.

$$Q(w+p) = Q(w) + [g(w)]^T p. \qquad (2.7)$$

Ponieważ funkcja Q(w) zależy od wag wyznaczonych w kroku t, natomiast Q(w+p) zależy od wag wyznaczonych w kroku t+1, więc aby zachodziła zależność $Q(w(t+1))<Q(w(t))$, wystarczy tak dobrać wektor p(t), żeby $g(w(t))^T p(t)<0$.

Można zauważyć, że warunek ten jest spełniony, jeśli przyjmiemy

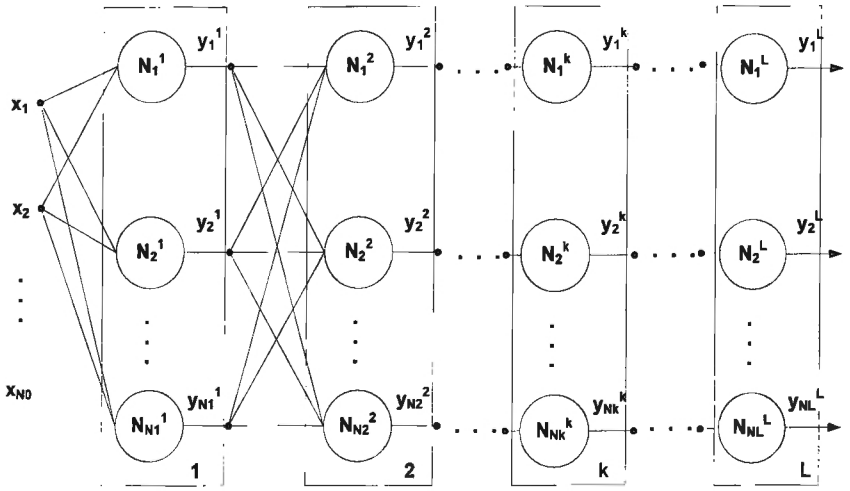
$$p(t) = -g(w(t)). \qquad (2.8)$$

Podstawiając zależność (2.17) do wzoru (2.15) otrzymujemy wzór określający sposób zmiany wag wielowarstwowej sieci neuronowej

$$w(t+1) = w(t) - \eta g(w(t)). \qquad (2.9)$$

Zależność (2.8) znana jest w literaturze pod nazwą reguły największego spadku.

W celu efektywnego wykorzystania zależności (2.8) do wyprowadzenia algorytmu wstecznej propagacji błędów, należy formalnie opisać schemat wielowarstwowej sieci neuronowej oraz wprowadzić odpowiednie oznaczenia. Na rys. 2 przedstawiono schemat wielowarstwowej sieci neuronowej.



Rys. 2. Wielowarstwowa sieć neuronowa

W każdej warstwie znajduje się N_k elementów, k=1,2,..., L, oznaczonych jako N_i^k, i=1,..., N_k. Elementy N_i^k będą nazywane neuronami, przy czym każdy z nich może być neuronem sigmoidalnym. Omawiana sieć neuronowa ma N_0 wejść, na które są podawane sygnały $x_1(t)$, ..., $x_{N0}(t)$ zapisywane w postaci wektora

$$x = [x_1(t),...,x_{N0}(t)]^T \qquad t=1,2,... \qquad (2.10)$$

Sygnał wyjściowy i-tego neuronu w k-tej warstwie jest oznaczony jako $y_i^{(k)}(t)$, i=1,..., N_k, k=1,2,..., L. Neuron N_i^k ma N_k wejść tworzących wektor

$$x^{(k)}(t) = [x_0^{(k)}(t),...,x_{Nk-1}^{(k)}(t)]^T, \qquad (2.11)$$

przy czym $x_i^{(k)}(t)=+1$ dla i=0 oraz k=1,..., L.

Sygnał wejściowy neuronu N_i^k jest powiązany z sygnałem wyjściowym warstwy k-1 w sposób następujący:

$$x_i^{(k)}(t) = \begin{cases} x_i(t) & dla \quad k=1, \\ y_i^{(k-1)}(t) & dla \quad k=2,...,L, \\ +1 & dla \quad i=0, k=1,...,L. \end{cases} \qquad (2.12)$$

Wektor wag neuronu N_i^k oznaczamy

$$w_i^{(k)}(t) = [w_{i,0}^{(k)}(t),...,w_{i,Nk-1}^{(k)}(t)]^T, \quad k=1,...,L, \; i=1,..., N_k. \qquad (2.13)$$

Sygnał wyjściowy neuronu N_i^k w chwili t, t=1,2,..., jest określony jako

$$y_i^{(k)}(t) = f(\varphi_i^{(k)}(t)), \qquad (2.14)$$

przy czym

$$\varphi_i^{(k)}(t) = \sum_{j=0}^{N_k} w_{ij}^{(k)}(t) x_j^{(k)}(t). \qquad (2.15)$$

Należy zauważyć, że sygnały wyjściowe neuronów w warstwie L-tej

$$y_1^L(t), y_2^L(t),..., y_{N_L}^L(t) \qquad (2.16)$$

są jednocześnie sygnałami wyjściowymi całej sieci. Są one porównywane z sygnałami wzorcowymi sieci

$$d_1^L(t), d_2^L(t),..., d_{N_L}^L(t). \qquad (2.17)$$

Błąd na wyjściu sieci Q jest definiowany następująco:

$$Q(t) = \sum_{i=1}^{N_L} \varepsilon_i^{(L)^2}(t) = \sum_{i=1}^{N_L} (d_i^{(L)}(t) - y_i^{(L)}(t))^2. \qquad (2.18)$$

Korzystając z zależności (2.7) i (2.16) otrzymujemy:

$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) - \eta \frac{\partial Q(t)}{\partial w_{ij}^{(k)}(t)}. \qquad (2.19)$$

W rezultacie algorytm wstecznej propagacji błędów można zapisać w postaci:

$$y_i^{(k)}(t) = f(\varphi_i^{(k)}(t)),\ \varphi_i^{(k)}(t) = \sum_{j=0}^{N_k} w_{ij}^{(k)}(t) x_j^{(k)}(t). \qquad (2.20)$$

$$Q_i^{(k)}(t) = \begin{cases} d_i^{(L)}(t) - y_i^{(L)}(t) & dla \quad k = L, \qquad (2.21) \\ \sum_{m=1}^{N_{k+1}} \delta_m^{(k+1)}(t) w_{mi}^{(k+1)}(t) & dla \quad k = 1,\ldots, L-1, \qquad (2.22) \end{cases}$$

$$\delta_i^{(k)}(t) = \varepsilon_i^{(k)}(t) f'(\varphi_i^{(k)}(t)), \qquad (2.23)$$

$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + 2\eta \delta_i^{(k)}(t) x_j^{(k)}(t). \qquad (2.24)$$

Przedstawione powyżej zależności matematyczne opisują sposób uczenia wielowarstwowej sieci neuronowej. Działanie algorytmu zaczyna się od podania wzorca uczącego na wejście sieci. Najpierw zostaje on przetworzony przez neurony warstwy pierwszej. Przetwarzanie oznacza wyznaczenie sygnału wyjściowego dla każdego neuronu w danej warstwie. Otrzymane w ten sposób sygnały stają się wejściami dla neuronów warstwy następnej. Cykl ten powtarza się aż do warstwy ostatniej. Znając sygnał wyjściowy warstwy ostatniej oraz sygnał wzorcowy z ciągu uczącego, można obliczyć błąd na wyjściu sieci zgodnie ze wzorem (2.21). Wykorzystując regułę delta, podobnie jak dla pojedynczego neuronu sigmoidalnego, można zmodyfikować wagi neuronów ostatniej warstwy, korzystając ze wzorów (2.21), (2.23), (2.24). Jednak w ten sposób nie można zmodyfikować wag w neuronach warstw ukrytych (nie znamy $\delta_i^{(k)}$ dla neuronów tych warstw), a przecież funkcja celu jest funkcją, w której zmiennymi są wszystkie wagi sieci. Dlatego błąd wyjściowy propagowany jest od warstwy wyjściowej do wejściowej zgodnie z połączeniami neuronów miedzy warstwami i z uwzględnieniem ich funkcji aktywacji (wzory (2.21), (2.23), (2.24)). Nazwa algorytmu pochodzi od sposobu jego realizacji, tzn. błąd jest „cofany" od warstwy wyjściowej do wejściowej.

Jak wspomniano wcześniej, uczenie sieci polega na znalezieniu minimum funkcji błędu. Jest to funkcja wielu zmiennych, którymi są wszystkie wagi. Funkcja taka może mieć wiele minimów lokalnych. Niestety, metoda największego spadku nie jest metodą odporną na występowanie minimów lokalnych i algorytm, poszukując rozwiązania, może w nich utknąć. Dlatego stosuje się modyfikacje algorytmu wstecznej propagacji błędów. Najprostszą metodą, która eliminuje problem minimów lokalnych, jest rozpoczynanie uczenia sieci neuronowej dla różnych wartości wag początkowych.

Inną metodą jest dodanie do standardowej postaci metody największego spadku dodatkowego współczynnika α zwanego momentum, zgodnie z którym wzór (2.24) modyfikujemy następująco:

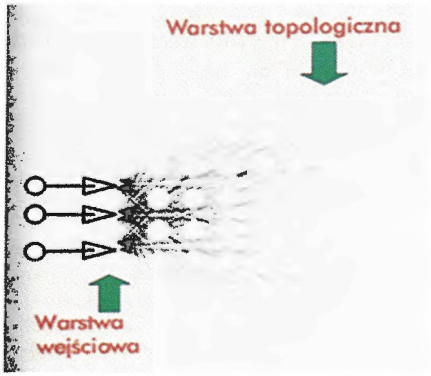
$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + 2\eta\delta_i^{(k)}(t)x_j^{(k)}(t) + \alpha[w_{ij}^{(k)}(t) - w_{ij}^{(k)}(t-1)]. \qquad (2.25)$$

Współczynnik ten uzależnia wagi w kroku następnym (t+1) nie tylko od ich wartości w kroku obecnym (jak w klasycznej metodzie wstecznej propagacji błędów), ale też w kroku poprzednim (t-1). Jeżeli w kolejnych iteracjach kierunek modyfikacji wagi jest ten sam, to składnik zawierający współczynnik momentum powoduje wzrost wartości przyrostu wagi i przesunięcie jej z większą siłą w kierunku minimum. Natomiast w przypadku odwrotnym składnik ten powoduje zahamowanie gwałtownych zmian wag. Współczynnik α przyjmuje wartości z zakresu (0,1), najczęściej zakłada się α=0,9.

2.2. Sieć neuronowa samoorganizująca się Kohonena [20]

Sieć neuronowa może być nauczana z nauczycielem lub może się uczyć sama. Sieć Kohonena nie tylko uczy się sama (bez nauczyciela), ale również tak uzgadnia (na zasadzie wzajemnych oddziaływań) funkcjonowanie wszystkich neuronów, że ich łączne, wypadkowe działanie stwarza pewną nową jakość. Chodzi o automatycznie tworzone przez sieć złożone odwzorowanie (mapping) zbioru sygnałów wejściowych w zbiór sygnałów wyjściowych. Odwzorowanie to na ogół nie daje się przewidzieć ani specjalnie zdeterminować, dlatego te sieci są znacznie bardziej samodzielne i niezależne niż np. sieci MLP, ponieważ ich właściwości są w niewielkim stopniu narzucone przez twórcę sieci, natomiast istotę ich działania oraz końcowe odwzorowanie, będące skutkiem ich pracy – determinuje głównie proces samoorganizacji.

Odwzorowanie jest to przekształcenie dowolnych sygnałów wejściowych we właściwe sygnały wyjściowe według określonych reguł. W przypadku wielu sygnałów wejściowych i wyjściowych to odwzorowanie staje się zadaniem bardzo trudnym, kłopotliwych i czasochłonnym. W takim przypadku pomocne mogą okazać się samoorganizujące się sieci neuronowe. Przykładowa struktura została przedstawiona na rys. 3.



Rys. 3. Struktura samoorganizującej się sieci neuronowej

Sieci samoorganizujące się mają zazwyczaj dużo wejść. Na rys. 3 pokazano jedynie 3 wejścia, żeby nie komplikować rysunku. Przy 30 wejściach plątanina połączeń byłaby niemożliwa do prześledzenia. W praktyce sieci tego rodzaju mają kilkanaście (a często nawet kilkadziesiąt) wejść, a ich użyteczność w kontekście różnych zastosowań jest z reguły tym większa, im więcej sygnałów wejściowych taka sieć potrafi obsłużyć. Jednak co ważniejsze i trudniejsze: sieci te mają zwykle jeszcze więcej wyjść, gdyż składają się z ogromnej liczby neuronów, tworzących łącznie tak zwaną warstwę topologiczną. Tu określenie „dużo neuronów" oznacza przynajmniej kilkadziesiąt, często kilkaset, nierzadko nawet kilka tysięcy neuronów – więc jest to dużo więcej niż w zwykłych sieciach MLP.

Jak zrozumieć, co właściwie sieć przedstawia w tej ogromnej warstwie topologicznej jako wynik swojej pracy?

Jest tutaj wykorzystana koncepcja rywalizacji neuronów oraz wyłaniania zwycięzcy. Po podaniu sieci dowolnego konkretnego sygnału wejściowego wszystkie neurony warstwy topologicznej obliczają swoje sygnały wyjściowe, będące odpowiedziami na ten sygnał. Wśród tych sygnałów jeden z reguły jest największy – i neuron, który właśnie ten największy sygnał wyprodukował staje się zwycięzcą (rys. 4).



Rys. 4. Przykładowy rozkład wartości sygnałów wyjściowych neuronów warstwy topologicznej oraz wyłoniony w wyniku oceny tych sygnałów „zwycięzca"

Proces samoorganizacji skutkuje między innymi tym, że taki zwycięski neuron jest tylko jeden oraz powoduje, że przewaga „zwycięzcy" nad „konkurentami" jest bardzo duża i wyraźna. Jedynie w przypadku, gdy sieć (po procesie samoorganizacji) dostanie na wejście jakiś zupełnie nietypowy sygnał (niepodobny do żadnego sygnału, który występował w zbiorze uczącym) – są trudności z wyłonieniem zwycięzcy, bo wszystkie neurony warstwy topologicznej produkują bardzo słabe (mało zróżnicowane) sygnały. Jest to jednak sytuacja nietypowa.

Na czym polega samoorganizacji w sieci i do czego może się przydać?

Sieci tworzą całkiem same, wyłącznie na podstawie obserwacji danych wejściowych, pewne odwzorowania zbioru sygnałów wejściowych w zbiór sygnałów wyjściowych, przy czym są to odwzorowania spełniające pewne ogólne kryteria, w żaden jednak sposób nie determinowane z góry przez twórcę sieci ani jej użytkownika. Struktura i właściwości potrzebnego nam odwzorowania powinny powstać same w następstwie procesu skoordynowanego samouczenia wszystkich elementów sieci. Takie spontaniczne tworzenie przez sieć potrzebnego odwzorowania sygnałów nazywane jest właśnie **samoorganizacją**.

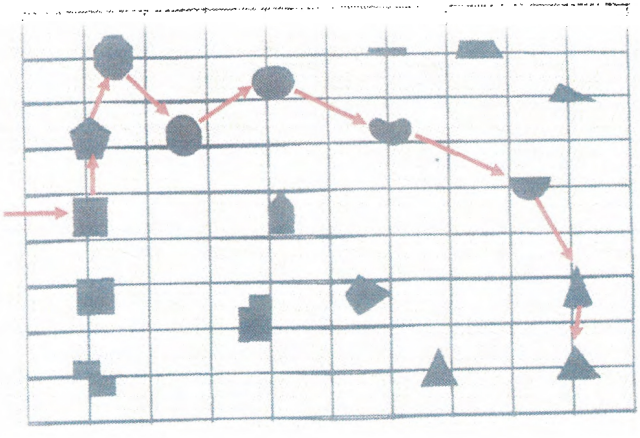
Z pewnego punktu widzenia jest to jedynie jeszcze jedna forma samouczenia, jeśli jednak popatrzymy na jej skutki to okaże się, że mamy tu do czynienia z wyraźnie wyższym stopniem adaptacji sieci, angażującym nie tylko optymalizację parametrów każdego neuronu z osobna, ale – co jest właśnie nowością – koordynację działań neuronów, wprowadzającą do obliczeń bardzo pożądane efekty **grupowania i kolektywności**.

Efekt **grupowania** polega na tym, że sieć w procesie samoorganizacji usiłuje rozdzielać dane wejściowe, wyróżniając wśród nich pewne **klasy podobieństwa**. Działa to tak, że wśród obiektów wejściowych (opisywanych przez wejściowe sygnały) wykrywane są (całkiem automatycznie) takie ich grupy, w których można umieścić sygnały podobne do siebie nawzajem i które jednocześnie są wyraźnie odmienne od sygnałów przypisanych do innych grup. Takie grupowanie danych jest bardzo przydatne w wielu zastosowaniach, wypracowano więc szereg specjalizowanych technik matematycznych pozwalających na analizowanie danych (najczęściej statystyczne) i na tworzenie właśnie takich grup danych. Te techniki nazywane są analizą skupień (cluster analysis). Są one chętnie stosowane w ekonomii – np. do wykrywania, które przedsiębiorstwa są podobne do siebie i mogą w związku z tym stwarzać podobne rokowania rentowności inwestycji. Są też stosowane w medycynie – do badania, które objawy wskazują na różne odmiany tej samej choroby, a które wskazują już na obecność nowej, być może nie znanej choroby.

Atrakcyjność neuronowego podejścia do problemu grupowania danych polega głównie na tym, że proces samoorganizacji może być tu prowadzony absolutnie automatycznie i spontanicznie, a twórca sieci nie musi jej przy tym udzielać żadnych wskazówek, ponieważ komplet potrzebnych informacji zawarty jest w samych wejściowych danych (z których sieć sama wydobędzie obserwację, że jedne są podobne do siebie nawzajem, a inne nie). Ponadto po nauczeniu sieci grupowania wejściowych danych powstaną bardzo użyteczne systemy – „zwycięskie" neurony, które wyspecjalizowały się w rozpoznawaniu poszczególnych klas sygnałów wejściowych, są więc ich detektorami i mogą być użyte do ich sygnalizowania.

Co to jest **kolektywność** działania sieci? Sieci, w których zachodzi samoorganizacja, są tak zorganizowane, że to, co rozpoznaje jeden neuron, w dużej mierze zależy także od tego, co rozpoznają inne neurony, znajdujące się w pobliżu. W ten sposób zbiorowość neuronów (czyli ich kolektywizm) może w sposób pełniejszy i bogatszy przetwarzać informacje niż każdy z neuronów z osobna wzięty.

Na rys. 5 pokazano efekt rozpoznawania figur geometrycznych w samoorganizującej się sieci neuronowej.



Rys. 5. Efekt rozpoznawania figur geometrycznych
w samoorganizującej się sieci neuronowej

Figury pokazano sieci jako obrazy. W miejscach, gdzie sieć ustawiła (całkiem sama!) neurony sygnalizujące poszczególne kształty pokazywanych jej figur geometrycznych – narysowano odpowiednie figury. To rozmieszczenie nie jest przypadkowe. Zgodnie ze strzałkami, które zostały dorysowane na rysunku należy zacząć od neuronu, który staje się „zwycięzcą", ilekroć sieci pokazany zostaje kwadrat. Niedaleko od niego jest neuron sygnalizujący pięciokąt, a nieco dalej – neuron sygnalizujący ośmiokąt. Słusznie, gdyż pięciokąt jest bardziej podobny do kwadratu niż ośmiokąt. Nieopodal ośmiokąta ulokował się neuron rozpoznający koło. Od niego krótka droga wiedzie do neuronu sygnalizującego elipsę, która z kolei jest niedaleko od neuronu rozpoznającego półkole. Półkole ma wyraźnie powinowactwo do mniejszego kawałka koła nazwanego sektorem, a ten z kolei ma wyraźnie zbliżony kształt do trójkąta...

Stąd wynika, że efekt kolektywnego działania sieci samoorganizujących się jest bardzo ciekawy. Ten efekt jest przyczynkiem do ogólniejszych rozważań: Otóż system, czyli właśnie zbiorowość odpowiednio powiązanych i współpracujących elementów, stwarza możliwość uzyskania nowych form zachowania i nowych postaci działań, znacznie bogatszych, niż by można oczekiwać biorąc pod uwagę każdy z elementów z osobna. Na przykład każdy pojedynczy owad jest dość prymitywnym zwierzęciem, a tymczasem zbiorowość owadów (rodzina pszczela, mrowisko) zdolne są do celowych, skomplikowanych i bez wątpienia inteligentnych działań.

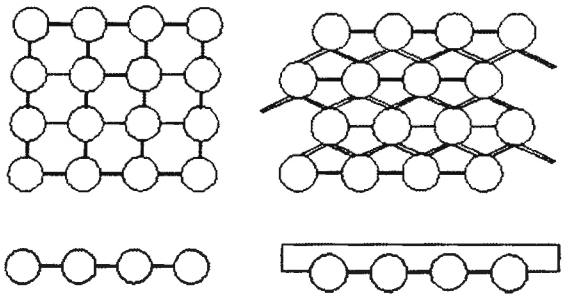
Kluczem do zrozumienia i używania sieci SOM jest pojęcie **sąsiedztwa neuronów**.

Jak wprowadza się do sieci sąsiedztwo?

Jeśli – na przykład- niektóre sygnały wejściowe są do siebie jakoś podobne (w dowolnym sensie), to może nam zależeć, żeby ich pojawianie się było sygnalizowane przez neurony sąsiadujące ze sobą w sieci. Tak działa sieć SOM, której wynik został pokazany na rys. 8.

Po raz pierwszy sąsiedztwo neuronów wykorzystał Kohonen i dlatego w literaturze sieci wykorzystujące sąsiedztwo nazwane są sieciami Kohonena. Do tej pory neurony sieci rozważane były jako jednostki w znacznym stopniu niezależne od siebie. Były one wprawdzie łączone ze sobą i przekazywały sobie nawzajem sygnały (jak to w sieci) – ale ich wzajemne **położenie** w warstwach nie miało żadnego znaczenia. Tymczasem w rozważanej sieci SOM bardzo ważne okazuje się to, które neurony warstwy topologicznej uznane zostaną za sąsiadujące ze sobą, gdyż będzie to w zasadniczy sposób rzutowało na zachowanie całej sieci.

Najczęściej neurony warstwy topologicznej kojarzymy z punktami jakiejś mapy, dlatego typowo rozpatrujemy sąsiedztwo dwuwymiarowe: neurony rozważa się tak, jakby były rozmieszczone w węzłach regularnej siatki złożonej z wierszy i kolumn. W siatce takiej każdy neuron ma przynajmniej czterech sąsiadów: dwóch w poziomie i dwóch w pionie. Jeżeli zachodzi taka potrzeba to sąsiedztwo można traktować szerzej: dopuścić także sąsiadów po przekątnej, albo dołączyć do sąsiedztwa neurony znajdujące się w dalszych rzędach lub kolumnach. Przykładowe topologie sieci, to siatka prostokątna, siatka heksagonalna, łańcuch otwarty i łańcuch zamknięty. Są one przedstawione na rys 6.



Rys 6. Topologie sieci: siatka prostokątna, siatka heksagonalna, łańcuch otwarty i łańcuch zamknięty. Linie łączące neurony określają relacje topologiczne

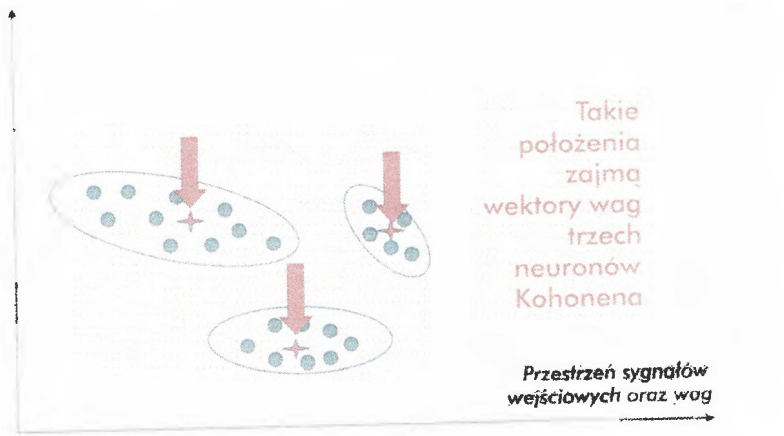
W specjalistycznych zastosowaniach sieci SOM można wprowadzić sąsiedztwo 3-, 4-, 5- i więcej wymiarowe. Jednak zdecydowanie najwięcej praktycznych zastosowań znajdują sieci jedno i dwuwymiarowe.

Sąsiedztwo dotyczy oczywiście wszystkich neuronów sieci: każdy neuron posiada komplet sąsiadów i sam z kolei jest sąsiadem dla innych neuronów. Tylko neurony znajdujące się na brzegu sieci nie mają kompletu sąsiadów, ale można temu zaradzić wprowadzając specjalną umowę (na przykład „domykając" sieć w taki sposób, że neurony z górnego brzegu są traktowane jako sąsiednie w stosunku do neuronów z dolnego brzegu; podobnie domyka się lewa i prawą krawędź sieci).

Co wynika z tego, że jakieś neurony uważa się za sąsiednie?

Fakt, że pewne neurony uznane są za sąsiednie, ma bardzo duże znaczenie. Kiedy w procesie uczenia jakiś neuron zostaje zwycięzcą i podlega procesowi uczenia – wraz z nim uczy się także jego sąsiadów. Jednak silnie uczony jest zwycięzca, a jego sąsiedzi są uczeni

nieco mniej intensywnie. Sąsiedzi są uczeni tak, aby w przyszłości mieli skłonność do rozpoznawania tego właśnie sygnału wejściowego. Jaki będzie skutek takiego uczenia? Otóż jeśli sygnały wejściowe do sieci będą nadchodziły w taki sposób, że będą wyraźnie wyróżnione ich skupiska, to poszczególne neurony sieci będą usiłowały zająć (poprzez swoje wektory wag) pozycje w centrach tych skupisk, zaś sąsiednie neurony będą „obstawiały" sąsiednie skupiska. Taką sytuację przedstawia rysunek 7, na którym kółkami zaznaczono punkty reprezentujące sygnały wejściowe, natomiast gwiazdki odpowiadają położeniu wektorów wag poszczególnych neuronów.



Rys. 7. Efekt uczenia sieci Kohonena, gdy sygnały wejściowe tworzą wyraźne skupiska

Gorsza sytuacja pojawi się wtedy gdy sygnały wejściowe będą równomiernie rozłożone w pewnym obszarze przestrzeni sygnałów wejściowych. Wówczas neurony będą miały tendencje do tego, by podzielić się funkcją rozpoznawania tych samych sygnałów w taki sposób, by każdy podzbiór sygnałów wejściowych miał wyróżniony neuron, który będzie wykrywał sygnały z innego podobszaru.

Po skończonej nauce neurony warstwy topologicznej podzielą pomiędzy siebie przestrzeń sygnałów wejściowych w taki sposób, że każdy obszar tej przestrzeni będzie sygnalizowany przez osobny neuron. Co więcej, te neurony, które zostały uznane za sąsiednie – będą wykazywały skłonność do rozpoznawania bliskich, czyli podobnych do siebie obiektów wejściowych.

Co potrafią w praktyce zrobić sieci Kohonena?

W istocie to co powstaje w sieci Kohonena jest to wewnętrzny model zewnętrznego świata. Tak samo jest z człowiekiem. Każdy ma neuron np. związany z identyfikowaniem drogi do domu, czy rozpoznający ulubione ciastko. Do każdego wrażenia zmysłowego, które rozpoznaje człowiek, do każdej sytuacji, którą zna – ma w mózgu model.

W przypadku działania robota, który ma dwa sensory, jeden sensor dostarcza informacji o wielkości oświetlenia, a drugi o natężeniu dźwięku. W ten sposób każdy punkt przestrzeni sygnałów wejściowych odpowiadać będzie jednemu środowisku o jakimś konkretnym zestawie cech – jedne będą jasne i ciche, inne ciemne i głośne, itp.

Robot wyposażony w sieć Kohonena zaczyna swoje działanie od tego, że obserwuje otoczenie. Czasem jest w tym otoczeniu jasno, czasem ciemno, czasem głośno, czasem cicho – ale okazuje się, że jedne kombinacje sygnałów wejściowych występują w otoczeniu robota – a inne nie. Robot klasyfikuje nadchodzące dane, gromadzi o nich wiedzę, wyspecjalizowuje swoje neurony – aż po pewnym czasie ma już wytrenowaną swoją sieć Kohonena, w której każdej zdarzającej się w praktyce sytuacji odpowiada neuron, który ją identyfikuje i wykrywa. Każdy neuron sieci Kohonena tego robota, rozpoznający jeden ze zdarzających się (tylko takich) stanów zewnętrznego świata, pełni w systemie sterowania robota funkcję gotowego wzorca. Zatem proces samouczenia sieci Kohonena tworzy w sterowniku robota taki swoisty zespół gotowych wzorców, będących zbiorek modeli dla każdej sytuacji, w jakiej może się znaleźć wykorzystujący sieć robot.

Taki model jest bardzo ważny, ponieważ mając model można z jego pomocą sklasyfikować każdą sytuację wejściową (wynikającą z sygnałów dostarczanych przez sensory). Z kolei zaklasyfikowawszy aktualną sytuację do jednej z wcześniej ustalonych klas, robot może odpowiednio dostosować do tej sytuacji swoje zachowanie. Np. jeśli sytuacja normalna (cicho i niezbyt jasno), to robot powinien posuwać się do przodu, jeśli światło zgaśnie to powinien się zatrzymać, itp. Ważne jest, że nie trzeba podawać szczegółowych wytycznych, co robot ma robić w każdej możliwej sytuacji. Na podstawie sąsiedztwa neuronów wykrywających podobne sytuacje można poprzestać na podaniu kilku wskazówek dla kilku wybranych sytuacji, a wytrenowana sieć Kohonena w „mózgu" robota sama ustali, co należy zrobić. Po prostu stwierdziwszy, że w stosunku do aktualnej sytuacji rejestrowanej przez jego sensory nie ma gotowej recepty postępowania, robot sam rozpozna (na podstawie struktury sąsiedztwa w sieci neuronowej), że spośród stanów sieci, dla których określono najwłaściwsze działanie, najbliższa jest pewna inna konkretna sytuacja. Jeśli dla tej najbliższej sytuacji zapisano w pamięci, że należy zrobić coś konkretnego – to zapewne w aktualnie rozpoznawanej podobnej sytuacji celowe jest zrobienie tego samego albo czegoś podobnego. Na przykład, jeśli sytuacja, stanowiąca wykorzystywaną analogię, związana była z poleceniem szybkiej jazdy do przodu – we wszystkich nie zdefiniowanych przez użytkownika sytuacjach „sąsiednich" można też zastosować jazdę do przodu, ale na przykład ze zredukowaną prędkością.

W ten sposób jeśli dla kilku tylko modelowych sytuacji określimy, co i kiedy należy robić – robot będzie umiał (lepiej lub gorzej) zachować się w dowolnej sytuacji. Warto podkreślić, że dotyczy to nie tylko ściśle i dokładnie tych obiektów, które pojawiły się w trakcie procesu samouczenia sieci. Podczas samouczenia sieci Kohonena występuje jak to zwykle w sieciach neuronowych proces uśredniania i generalizacji.

Sieć Kohonena i jego algorytm

Sieć Kohonena została nazwana przez jego twórcę samoorganizującym się odwzorowaniem (Self-Organizing Map - SOM) lub samoorganizującym się

odwzorowaniem cech (Self-Organizing Feature Map - SOFM) i jest najbardziej popularnym typem sieci określanej mianem sieci samoorganizującej się.

W sieci tej mamy do czynienia z uczeniem konkurencyjnym. Oznacza to, że użytkownik ma do dyspozycji jedynie wzorce wejściowe, nie posiada natomiast żadnych wzorców wyjścia. Zadaniem sieci w trakcie procesu uczenia jest wytworzenie takich wzorców. Sieć ma na celu utworzenie takiej struktury, która w najlepszy sposób będzie odwzorowywała zależności w przestrzeni wektorów wejściowych.

Sieć jest zwykle jednokierunkowa, w której każdy neuron jest połączony ze wszystkimi składowymi N-wymiarowego wektora wejściowego x. Wagi połączeń neuronów tworzą wektor $w_i=[w_{i1}, w_{i2}, ..., w_{iN}]^T$. Wektor sygnałów wejściowych x jest przed procesem uczenia normalizowany $\|x\|=1$, co można zapisać jako

$$x_i = \frac{x_i}{\sqrt{\sum_{r=1}^{N}(x_r)^2}} \tag{2.26}$$

Po pobudzeniu sieci wektorem wejściowym x, podczas współzawodnictwa wygrywa neuron, którego wagi najmniej różnią się od odpowiednich składowych tego wektora. Zwycięzca, neuron w-ty, spełnia relację

$$d(x, w_w) = \min_{1 \le i \le n} d(x, w_i) \tag{2.27}$$

gdzie d(x, w) oznacza odległość w sensie wybranej metryki między wektorem n a wektorem w.

Wokół i-tego neuronu przyjmuje się topologiczne sąsiedztwo G(i,x).

W klasycznym algorytmie Kohonena funkcja G(i,x) jest definiowana następująco:

$$G(i,x) = \begin{cases} 1 & dla \quad d(i,w) \le R \\ 0 & dla \quad pozostałych \end{cases} \tag{2.28}$$

gdzie d(i,w) oznacza odległość euklidesową między neuronem zwycięskim w_i a i-tym neuronem, R – promieniem sąsiedztwa.

Współczynnik uczenia jest promieniem sąsiedztwa o wartościach malejących z czasem. Ten typ, to sąsiedztwo prostokątne.

Drugim typem sąsiedztwa, z jakim można się spotkać w mapach Kohonena, jest sąsiedztwo gaussowskie, gdzie

$$G(i,x) = exp\left(-\frac{d^2(i,w)}{2R^2}\right) \tag{2.29}$$

O stopniu adaptacji neuronów z sąsiedztwa zwycięzcy decyduje odległość euklidesowa oraz promień sąsiedztwa. Przy sąsiedztwie gausowskim stopień adaptacji jest zróżnicowany, w przeciwieństwie do sąsiedztwa prostokątnego, gdzie każdy neuron podlega adaptacji w tym samym stopniu i zależy od funkcji Gaussa. Sąsiedztwo gausowskie jest lepsze i prowadzi do lepszych rezultatów uczenia i lepszej organizacji sieci.

Wagi neuronów modyfikuje się zgodnie z zależnością:

$$w_i(n+1) = w_i(n) + \eta G(i,x)[x - w_i(n)] \qquad (2.30)$$

Gdzie: w_i oznacza wektor wag w i-tym neuronie, n – numer kroku iteracji, η - współczynnik uczenia

We wzorze tym oddzielono współczynnik uczenia η każdego neuronu (i) od jego odległości względem wektora x, uwzględnionej przez funkcję sąsiedztwa G(i,x).

W mapach odwzorowań topologicznych Kohonena najpierw określany jest zwycięzca, stosując euklidesową miarę odległości, a następnie wartość współczynnika uczenia neuronów należących do sąsiedztwa zwycięzcy.

Miara odległości

W procesie samoorganizacji na każdym etapie należy wyłonić zwycięzcę - neuron, którego wektor wagowy różni się najmniej od wektora na wejściu x. Aby móc określić, która odległość jest najmniejsza, należy dobrać odpowiednią metrykę, w jakiej mierzona będzie odległość między wektorem x a w_i. Najczęściej używa się

- miary euklidesowej

$$d(x,w_i) = \|x - w_i\| = \sqrt{\sum_{j=1}^{N}(x_j - w_{ij})^2} \qquad (2.31)$$

- iloczynu skalarnego

$$d(x,w_i) = 1 - xw_i = 1 - \|x\|\|w_i\|\cos(x,w_i) \qquad (2.32)$$

- miary wedlug normy L1 (Manhattan)

$$d(x,w_i) = \sqrt{\sum_{j=1}^{N}|x_j - w_{ij}|} \qquad (2.33)$$

- miary według normy L

$$d(x, w_i) = \max_j |x_j - w_{ij}| \tag{2.34}$$

Najciekawsze efekty uzyskuje się, wprowadzając metody Kohonena dla sieci neuronowych dwuwymiarowych, to znaczy takich, w których neurony ułożone są w strukturę tablicy o pewnej liczbie wyróżnionych wierszy i kolumn. Dla tego typu sieci opisane są w literaturze bardzo interesujące formy zachowania. Sieci tego typu są w stanie samorzutnie się organizować, by odwzorować strukturę wejściowego dwuwymiarowego obiektu - na przykład samorzutnie utworzyć siatkę równomiernie pokrywającą określony kształt analizowanego obszaru.

Etapy działania samoorganizujących się sieci neuronowych

Funkcjonowanie samoorganizujących się sieci neuronowych odbywa się w trzech etapach:

- konstrukcja
- uczenie
- rozpoznawanie.

System, który miałby realizować funkcjonowanie sieci samoorganizującej się, powinien składać się z kilku podstawowych elementów. Pierwszym z nich jest macierz neuronów pobudzanych przez sygnały wejściowe. Sygnały te powinny opisywać pewne charakterystyczne cechy zjawisk zachodzących w otoczeniu tak, aby na ich podstawie sieć była w stanie je pogrupować. Informacja o zdarzeniach jest przekładana na bodźce pobudzające neurony. Zbiór sygnałów przekazywanych do każdego neuronu nie musi być identyczny, nawet ich ilość może być różna. Muszą one jednak spełniać pewien warunek, a mianowicie jednoznacznie określać dane zdarzenia.

Kolejną częścią składową sieci jest mechanizm, który dla każdego neuronu określa stopień podobieństwa jego wag do danego sygnału wejściowego oraz wyznacza jednostkę z największym dopasowaniem - zwycięzcę. Obliczenia zaczynamy dla wag równych małym liczbom losowym, przy czym ważne jest, aby nie zachodziła żadna symetria. W trakcie uczenia wagi te są modyfikowane w taki sposób, aby najlepiej odzwierciedlać wewnętrzną strukturę danych wejściowych. Istnieje jednak niebezpieczeństwo, że zwiążą się one z pewnymi wartościami, zanim jeszcze grupy zostaną prawidłowo rozpoznane, i wtedy trzeba ponawiać uczenie z innymi wagami.

Wreszcie konieczne do przeprowadzenia samoorganizacji jest, aby sieć była wyposażona w zdolność do adaptacji wartości wag neuronu zwycięzcy i jego sąsiadów w zależności od siły, z jaką odpowiedział on na dane wejście. Topologię sieci można w łatwy sposób określić poprzez zdefiniowanie sąsiadów dla każdego neuronu. Załóżmy, że jednostkę, której odpowiedź na dane pobudzenie jest maksymalna, będziemy nazywali "obrazem" tego pobudzenia. Wtedy możemy przyjąć, że sieć jest uporządkowana, jeśli topologiczne relacje między sygnałami wejściowymi i ich obrazami są takie same.

2.3. Podsumowanie sieci neuronowych użytych do eksperymentów

Sieci MLP niezmiennie są najbardziej rozpowszechnionymi i uniwersalnymi sieciami neuronowymi stosowanymi do rozwiązywania różnych problemów.

Samoorganizujące się mapy, zwane też sieciami Kohonena, to sieci neuronów, z którymi są stowarzyszone ich współrzędne definiowane na prostej, płaszczyźnie lub w dowolnej *n*-wymiarowej przestrzeni. Uczenie tego rodzaju sieci polega na zmianach współrzędnych neuronów tak, aby dążyły one do wzorca zgodnego ze strukturą analizowanych danych. Oznacza to, że sieci „rozpinają się" wokół zbiorów danych, dopasowując do nich swoją strukturę.

3. Eksperymenty dotyczące tworzenia algorytmów lokalizacji awarii w sieci wodociągowej

3.1. Stan obecny

W Instytucie Badań Systemowych PAN opracowano do tej pory 2 algorytmy lokalizowania awarii w sieci wodociągowej. Pierwszy z nich korzysta jedynie z systemu monitoringu i został częściowo przetestowany w wodociągach rzeszowskich [16]. Polega on na tym, że dla punktów pomiarowych zainstalowanych na sieci wodociągowej wyznacza się, na podstawie pomiarów historycznych, tak zwane standardowe względnie wzorcowe krzywe przepływów godzinowych i porównuje się je z bieżącymi pomiarami odczytywanymi przez system monitoringu. Jeżeli różnica między wyznaczaną krzywą bieżącą i wyznaczoną krzywą wzorcową jest mniejsza od zadanego przedziału tolerancji, to przyjmuje się, że sieć wodociągowa pracuje bezawaryjnie. Jeżeli natomiast ta różnica jest duże, to przyjmuje się, że w otoczeniu punktu pomiarowego sygnalizującego tą różnicę nastąpił stan awaryjny.

Z kolei inny algorytm lokalizacji awarii korzysta już zarówno z systemu monitoringu, jak i skalibrowanego modelu hydraulicznego sieci wodociągowej [17]. Algorytm ten składa się z następujących kroków:

- Wykonanie obliczeń hydraulicznych sieci, dla zadanego poziomu obciążenia, za pomocą modelu hydraulicznego.
- Zapis wartości przepływów i ciśnień w punktach pomiarowych systemu monitoringu dla wykonanych obliczeń (zapis standardowego rozkładu ciśnień i przepływów).
- Symulacja wycieków w kolejnych węzłach sieci wodociągowej za pomocą modelu hydraulicznego.
- Zapis wartości przepływów i ciśnień w punktach pomiarowych dla wykonanych symulacji stanów awaryjnych (zapis awaryjnych rozkładów ciśnień i przepływów).
- Rejestracja pomiaru z systemu monitoringu, odbiegającego od standardu dla zadanego poziomu obciążenia sieci.
- Zapis wartości przepływów i ciśnień w punktach pomiarowych dla stwierdzonej nietypowej sytuacji (zapis nietypowego rozkładu ciśnień i przepływów).

- Znalezienie wśród zapisów stanów awaryjnych zapisu najbardziej zbliżonego do zapisu nietypowego. Znaleziony zapis wskazuje węzeł sieci, w którym najprawdopodobniej doszło do wycieku.

3.2. Analiza lokalizacji awarii przy wykorzystaniu sieci neuronowej

Analiza obecnego stanu badań dotyczących wspomagania wykrywania i lokalizacji wycieków wody w sieci wodociągowej wykazała, że jest miejsce na kolejne badania dotyczące tego problemu. W ramach projektu opracowywane są algorytmy lokalizacji awarii w postaci sieci neuronowych dla sieci wodociągowej na podstawie danych rzeczywistych z Rzeszowa. Do modelowania sieci wodociągowej, dla celów porównawczych i weryfikacji wyników, zostaną zastosowane dwa typu sieci neuronowe: MLP oraz Kohonena.

3.2.1. Algorytm lokalizacji awarii sieci wodociągowej

Pierwszy etap prac z wykorzystaniem sieci neuronowych dotyczy opracowania samego algorytmu lokalizacji awarii sieci wodociągowej, który w przyszłości stanie się elementem tworzonego systemu komputerowego [24]. Bazując na określonej liczbie czujników przepływu zainstalowanych w przewodach sieci wodociągowej i dysponując odpowiednio wytrenowaną siecią neuronową, można pojawiające się awarie sieci szybko i w sposób automatyczny wykrywać i lokalizować.

Algorytm ten składa się z następujących kroków:

1. Wyznaczenie punktów wrażliwych sieci wodociągowej przy użyciu systemu do optymalnego planowania systemu monitoringu.
2. Wybór odpowiedniej liczby najbardziej wrażliwych punktów pomiarowych, z których pozyskuje się największy zasób informacji o występujących w sieci stanach awaryjnych.
3. Wykonanie obliczeń hydraulicznych sieci, dla zadanego poziomu obciążenia, za pomocą wybranego i odpowiednio skalibrowanego modelu hydraulicznego.
4. Zapis wartości przepływów i ciśnień w punktach pomiarowych systemu monitoringu dla wykonanych obliczeń (zapis standardowego rozkładu ciśnień i przepływów).
5. Symulacja wycieków w kolejnych węzłach sieci wodociągowej za pomocą modelu hydraulicznego.
6. Zapis wartości przepływów i ciśnień w punktach pomiarowych dla wykonanych symulacji stanów awaryjnych (zapis awaryjnych rozkładów ciśnień i przepływów).
7. Utworzenie klasyfikatora lokalizującego awarie w postaci różnego typu sieci neuronowych i wybór klasyfikatora najlepszego według kryterium największej wrażliwości.
8. Rejestracja z systemu monitoringu pomiaru odbiegającego od standardu dla zadanego poziomu obciążenia sieci, czyli w przypadku wystąpienia awarii znalezienie wśród modeli sieci neuronowych modelu takiego, który wskazuje dokładnie węzeł sieci z zaistniałym wyciekiem.

3.2.2. Wyznaczenie punktów wrażliwych sieci wodociągowej przy użyciu algorytmu do planowania systemu monitoringu

W celu znalezienia najlepszych lokalizacji czujników w punktach pomiarowych systemu monitoringu wyznacza się tzw. punkty wrażliwe sieci wodociągowej przy użyciu algorytmu do planowania systemu monitoringu.

Aby zmaksymalizować ilość informacji przekazywanych przez punkty pomiarowe, powinny on być tak wybrane, aby przekazywały wiedzę o stanie sieci nie tylko z danego punktu, ale również z jego także dalszego otoczenia. Takie punkty charakteryzujące się dużą wrażliwością na zmiany stanu sieci nazwiemy punktami charakterystycznymi. Maksymalizacja przekazywanej informacji oznacza zwykle, w tradycyjnie planowanych systemach monitoringu, zwiększanie liczby punktów pomiarowych, a więc działanie przeciwne do opisanego powyżej.

Dlatego należy podkreślić, że odpowiedni wybór stosunkowo niewielkiej liczby punktów charakterystycznych może być równoważny pod względem jakości i ilości przekazywanej wiedzy o sieci większej liczbie punktów zlokalizowanych w mniej wrażliwych miejscach sieci. Wybór odpowiednich punktów pomiarowych do monitorowania sieci wodociągowej jest nietrywialnym zadaniem, do rozwiązania którego można stosować różne algorytmy obliczeniowe. Jednym z nich jest algorytm podany przez R. Straubela i B. Holznagela.

Przy tym jako miarę wrażliwości sieci na symulowane wycieki można stosować na przykład sumę różnic bezwzględnych między pomiarami ciśnień dla standardowego stanu pracy sieci i stanów awaryjnych dla przyjmowanych różnych kombinacji liczby i lokalizacji punktów pomiarowych, w przypadku, gdy w systemie monitoringu uwzględniamy jedynie pomiary ciśnienia.

Następnym etapem algorytmu jest dodanie punktów pomiarowych przepływu.

Punkty pomiarowe zostały wyznaczone w miejscach o największej wrażliwości na wycieki. Eksperymenty zostały przeprowadzone dla dwóch przypadków. Pierwszy zawierał 10 punktów pomiarowych, a drugi 20. Zwiększenie liczby punków pomiarowych spowodowało zwiększenie dokładności lokalizacji awarii w sieci.

Na początku wartość pomiaru w punktach monitoringu jest 0.

3.2.3. Wykonanie obliczeń hydraulicznych sieci, dla zadanego poziomu obciążenia, za pomocą wybranego modelu hydraulicznego

W celu pozyskania danych do kalibracji modelu hydraulicznego sieci wodociągowej i w dalszej kolejności do opracowania algorytmu lokalizacji awarii wykorzystuje się system monitoringu zainstalowany na sieci. Współcześnie w coraz większym stopniu nadzorowanie parametrów sieci wodociągowej odbywa się z wykorzystaniem systemów monitoringu SCADA. Przepływy i ciśnienia wody określają stan sieci, więc bieżące informacje o wartościach tych parametrów pozwalają zorientować się, czy sieć pracuje właściwie.

Monitoring sieci wodociągowej służy do zbierania, archiwizacji i wizualizacji historycznych i bieżących informacji o pracy i stanie sieci.

Wykonanie obliczeń hydraulicznych sieci, zapis wartości przepływów i ciśnień w punktach pomiarowych systemu monitoringu dla wykonanych obliczeń (zapis standardowego rozkładu ciśnień i przepływów), symulacja wycieków w kolejnych węzłach sieci wodociągowej za pomocą modelu hydraulicznego w miejscach wrażliwych oraz zapis wartości przepływów i ciśnień w punktach pomiarowych dla wykonanych symulacji stanów awaryjnych (zapis awaryjnych rozkładów ciśnień i przepływów) służą do przygotowania plików uczących dla sieci neuronowych, które posłużą z kolei jako modele do lokalizacji awarii w sieci wodociągowej.

Oznacza to, że odpowiednio skalibrowany model hydrauliczny sieci wodociągowej jest traktowany w obliczeniach jako obiekt rzeczywisty, natomiast sieć neuronowa jest traktowana jako model obiektu. Natomiast do kalibracji modelu hydraulicznego służą dane pomiarowe z rzeczywistego obiektu, czyli z badanej sieci wodociągowej.

Wykonanie obliczeń hydraulicznych sieci powoduje pojawienie się określonych wartości pomiaru w punktach pomiarowych. Są one wizualizowane na ekranie modelu hydraulicznego sprzęgniętego z systemem monitoringu.

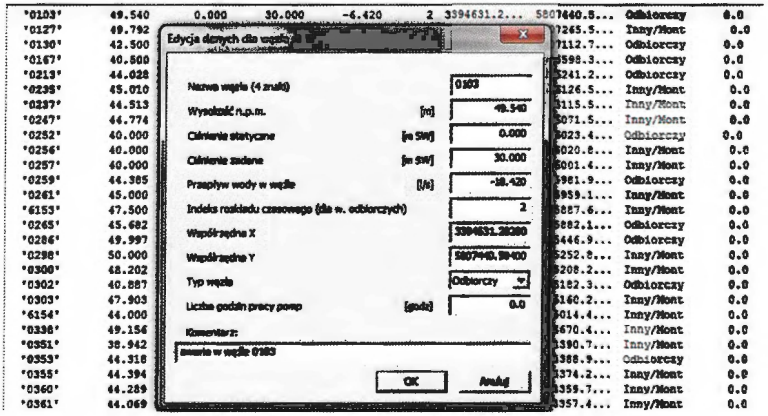
Model hydrauliczny sieci wodociągowej o nazwie MOSUW-H opracowano w Instytucie Badań Systemowych PAN jako element zintegrowanego systemu informatycznego do kompleksowego zarządzania miejskim systemem zaopatrzenia w wodę.

W ten sposób dokonano zapisu standardowego rozkładu przepływów oraz wykonano szereg symulacji wycieków w różnych węzłach sieci wodociągowej za pomocą modelu hydraulicznego w miejscach wrażliwych.

Symulację awarii wykonano w 4 krokach:

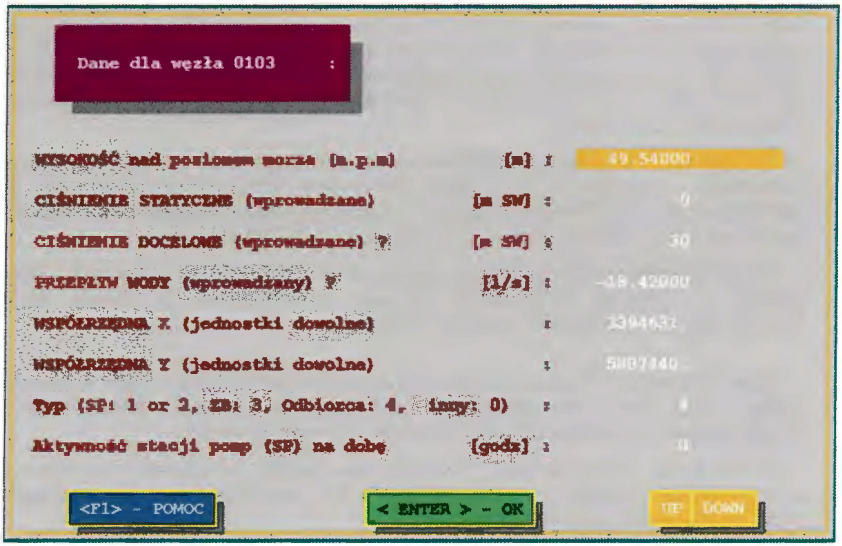
1. **Zwiększono kilkakrotnie wartość przepływu wody w wybranym węźle**

Na rysunku 8 pokazano przykładową symulację awarii węzła 0103 poprzez zmianę przepływu wody z wartości -6,420 na wartość -18,420 [l/s].



Rys. 8. Symulacja awarii węzła 0103

Rysunek 9 pokazuje zmienione dane dla węzła 0103.



Rys. 9. Dane węzła 0103

2. Obliczono hydraulikę sieci wodociągowej z awarią

Dla każdej symulowanej awarii zostały w ten sposób zmienione dane wejściowe. Po zmianie danych wejściowych wykonano obliczenia hydrauliczne sieci wodociągowej oraz dokonano zapisu wartości przepływów w punktach pomiarowych systemu monitoringu dla wykonanych symulacji stanów awaryjnych (zapis awaryjnych rozkładów przepływów).

3. Obliczono różnicę przepływów w sieci w normie i stanie awarii

Przykład dla 10 punktów pomiarowych

id węzła	przepływ norma	przepływ awaria	różnica przepływów
PPP001	-0,066	-0,053	-0,013
PPP002	-0,066	-0,053	-0,013
PPP003	0,588	-2,026	2,614
PPP004	-0,768	-0,275	-0,493
PPP005	0,035	0,043	-0,008
PPP006	0,166	0,134	0,032
PPP007	0,166	0,134	0,032
PPP008	2,655	1,043	1,612

PPP009	0,886	2,023	-1,137
PPP010	-0,886	-2,023	1,137

Przykład dla 20 punktów pomiarowych

id węzła	przepływ norma	przepływ awaria	różnica przepływów
PPP001	-0,066	-0,053	-0,013
PPP002	-0,066	-0,053	-0,013
PPP003	0,588	-2,026	2,614
PPP004	-0,768	-0,275	-0,493
PPP005	0,035	0,043	-0,008
PPP006	0,166	0,134	0,032
PPP007	0,166	0,134	0,032
PPP008	2,655	1,043	1,612
PPP009	0,886	2,023	-1,137
PPP010	-0,886	-2,023	1,137
PPP011	-2,594	-1,715	-0,879
PPP012	3,098	4,7	-1,602
PPP013	-4,64	-3,687	-0,953
PPP014	2,244	1,107	1,137
PPP015	0,676	1,152	-0,476
PPP016	-0,472	-0,576	0,104
PPP017	1,866	1,853	0,013
PPP018	-0,858	-0,577	-0,281
PPP019	1,746	2,883	-1,137
PPP020	1,072	1,082	-0,01

4. Określono punkt pomiarowy, który był najbardziej wrażliwy na awarię węzła

Jeśli awaria wywołała w jakimś punkcie największą zmianę przepływu, to to sugeruje, że ten punkt pomiarowy jest najbliżej awarii i przez to lokalizujemy awarię. Oczywiście sieć wodociągowa jest układem zamkniętym i inne punkty pomiarowe też jakoś zareagują bardziej lub mniej. To widać w tabelkach w punkcie 3.

Przykład dla 10 punktów pomiarowych

id węzła	przepływ norma	przepływ awaria	różnica przepływów
PPP003	0,588	-2,026	2,614

Przykład dla 20 punktów pomiarowych

PPP003	0,588	-2,026	2,614

W ten sposób dokonano wielu symulacji awarii sieci wodociągowej w różnych węzłach w celu przygotowania plików uczących dla sieci neuronowych, które posłużą z kolei jako modele do lokalizacji awarii w sieci wodociągowej.

Mając dane wartości przepływu i ciśnienia wody generowane przez model hydrauliczny sieci, skalibrowany na podstawie rzeczywistych pomiarów, można przystąpić do budowy klasyfikatora w postaci sieci neuronowej, który będzie sygnalizował wystąpienie awarii w sieci wodociągowej oraz podawał jej lokalizację.

3.2.4. Tworzenie klasyfikatora

Wstępne przetwarzanie danych dla sieci neuronowej

Wstępne przetwarzanie danych odgrywa ważną rolę zarówno podczas uczenia, jak i testowania sieci neuronowych. Na tym etapie należy rozwiązać takie problemy jak właściwy dobór cech, ich selekcja czy wybór odpowiednich przykładów.

Z właściwym doborem cech wiążą się pewne dodatkowe problemy ograniczające wybór optymalnego zbioru cech, takie jak problem wymiarowości, skorelowania danych, czy współzależności danych. Problem wymiarowości polega na konieczności dostarczenia dostatecznej liczby przykładów uczących. Innego rodzaju problemem są współzależności występujące pomiędzy cechami polegające na tym, że pewne cechy wnoszą istotne informacje tylko wówczas, gdy rozpatruje się je razem. W trzecim problemie w przypadku silnie skorelowanych zmiennych, rozwiązaniem może być wybór pewnego podzbioru cech, w którym pomiędzy cechami nie będą już występować korelacje. Silne korelacje między zmiennymi powodują pogorszenie jakości informacji otrzymanej przez sieć, co ma istotny wpływ na proces uczenia.

Kolejnym problemem związanym ze wstępnym przetwarzaniem danych jest reprezentacja wartości symbolicznych przez sieć neuronową. Sieć neuronowa przetwarza jedynie wartości numeryczne. Dlatego do budowy modelu zastosowano sieć wspomaganą układami kodującymi i klasyfikującymi. W układzie kodowania zastosowano kodowanie 1 z n. Numery punktów pomiarowych są wielkościami jakościowymi.

W związku z zamianą wartości symbolicznych na numeryczne rośnie liczba wejść i wyjść sieci neuronowej.

Identyfikacja struktury i parametrów wybranego modelu sieci

Identyfikacja struktury modelu, daje odpowiedź:
1. ile warstw zawiera sieć neuronowa (czy potrzebne są warstwy ukryte ?)
2. ile neuronów występuje w każdej warstwie ?
3. czy należy dołączyć dodatkowy neuron w celu lepszej stabilności sieci w trakcie uczenia (bias) ?
4. jak dobrać parametry uczenia sieci neuronowej (współczynnik uczenia, współczynnik momentu, itp.) ?

Współczynnik uczenia

Przyjmuje on wartości z przedziału od 0 do 1, nie licząc zera (może być większy od 1, choć nie ma to większego sensu). Obliczona korekta jest mnożona przez ów współczynnik. Jeśli ma on wysoką wartość (około 1), korekta nie zmienia się znacząco. Jeśli jednak współczynnik ma wartość na przykład 0.1, korekta zostanie dziesięciokrotnie zmniejszona. Współczynnik uczenia (w angielskiej literaturze *learning rate*) można porównać do nauczyciela: zbyt wysoka wartość odpowiada nauczycielowi surowemu, który nawet za małe pomyłki stosuje wysoką karę. Uczeń (czyli sieć) pełen frustracji miota się od jednej skrajności do drugiej i nie może trafić na właściwe rozwiązanie. Zbyt niski współczynnik z kolei to nauczyciel zbyt łagodny, przy którym uczeń uczy się bardzo powoli, gdyż jego błędy są tolerowane.

Współczynnik momentu

W celu zwiększenia efektywności procesu uczenia opracowano momentową metodę wstecznej propagacji błędu, która do klasycznej idei optymalizacji gradientowej wprowadziła element bezwładności kolejnych zmian wektora wag. Algorytm ten w rzeczywistości wykonuje wygładzanie poprawek zbioru wag, uzależnione od wartości współczynnika wygładzania określanego nazwą momentu (ang. momentum). Wpływ wcześniejszych poprawek wektora wag maleje wykładniczo w miarę oddalania się od bieżącego numeru iteracji. Ostateczna poprawka jest dzięki temu sumą opartą nie tylko o wielkość bieżącego gradient funkcji celu, ale także o wcześniejsze modyfikacje.

Podobnie jak w przypadku współczynnika uczenia, także współczynnik momentu w najprostszej postaci algorytmu wprowadzany jest jako wartość stała, niezmienna podczas trwania treningu sieci neuronowej.

Identyfikacja parametrów modelu polega na dobraniu wag połączeń między neuronowych. Wagi połączeń dobierane są tak długo, aż wartość błędu średniokwadratowego nie osiągnie minimalnej wartości, co pozwoli na zakończenie procesu uczenia.

Liczba warstw ukrytych oraz neuronów w tych warstwach mają istotny wpływ na jakość działania sieci warstwowej. Liczba neuronów ukrytych najczęściej dobierana jest eksperymentalnie. Problem jest dosyć złożony: zbyt mała liczba neuronów ukrytych powoduje niezdolność sieci do zgromadzenia dostatecznej wiedzy o rozwiązywanym problemie, a zbyt bogata architektura jest z kolei przyczyną upośledzenia tzw. generalizacji (sieć zapamiętuje zbyt dokładnie dane treningowe, co powoduje kłopoty z uogólnieniem ich na przypadki nie objęte procesem uczenia).

Problem ten rozwiązywany jest zwykle na drodze eksperymentalnej. Wielokrotne powtarzanie procesu uczenia pozwala zaprojektować sieci dostatecznie duże, by mogły nauczyć się problemu, a jednocześnie na tyle małe, aby poprawnie generalizować.

Weryfikacja modelu sieci neuronowej

Weryfikacja modelu sieci polega na testowaniu, walidacji oraz korekcie modelu sieci neuronowej. Weryfikacja modelu zawiera testowanie, walidację i korektę.

Testowanie modelu sprawdza działanie sieci na danych mieszczących się w pliku testowym. Dane testowe to zbiór przypadków, które nigdy nie były używane do uczenia danej sieci. Są one natomiast używane, poprzez badanie dokładności predykcji, do oceny

postępów sieci w coraz lepszym modelowaniu zależności wejście-wyjście, w miarę uczenia sieci. Jest przeprowadzone z wykorzystaniem pliku z danymi testowymi.

Walidacja modelu pozwala sprawdzić, czy dla nowych wartości wejściowych sieć poprawnie dobiera parametry wyjściowe. Walidacja sprawdza dokładność sieci neuronowej na podstawie nowego zestawu danych, które nie były wykorzystywane w procesie uczenia sieci. Jest przeprowadzona z wykorzystaniem pliku walidacyjnego.

Korekta modelu polega na zmianie ilości neuronów w sieci, douczeniu sieci, jeśli sieć jest zbyt dokładnie nauczona, lub też zmianie liczby warstw ukrytych. Korekta modelu jest wykonywana w przypadku, gdy sieć daje niezadawalające wyniki. Jest zbyt mało dokładna. Tak naprawdę to korekta daje już nowy model.

W celu oszacowania jakości wiedzy, jaką sieć zdobyła na etapie uczenia, rozpatrywane mogą być cztery wskaźniki: błąd średniokwadratowy RMS, tolerancja uczenia, tolerancja testowania oraz tolerancja walidacji. Tolerancja uczenia, testowania i walidacji służy do określenia dopuszczalnego błędu na wyjściu sieci. Tolerancja jest pojęciem odwrotnym do jakości.

Jakość dla podzbioru uczącego, testowego i walidacyjnego została obliczona jako proporcja poprawnie zaklasyfikowanych przypadków w stosunku do wszystkich przypadków ze zbioru.

Użytkownik ma możliwość śledzenia tych wskaźników, co pozwala mu na podjęcie decyzji o zakończeniu uczenia. Błąd RMS jest podstawową miarą błędu, który jest obliczany jako suma kwadratów różnic wartości rzeczywistej i oczekiwanej, podzieloną przez liczbę wyrazów. Mankamentem tej miary jest fakt, że nie odzwierciedla ona błędu pojedynczego wyjścia sieci, lecz pozwala na stwierdzenie średniej odchyłki w ramach okresu prezentacji wzorców (epoki). Dlatego dodatkowo sprawdzany jest wskaźnik tolerancji, który określa dopuszczalny błąd na pojedynczym wyjściu sieci.

Rzeczywista wartość to jest wartość uzyskana na wyjściu sieci. Oczekiwana to jest wartość wzorcowa.

Epoka – jednokrotne powtórzenie iteracyjnej procedury uczenia sieci neuronowej, w trakcie której tej sieci prezentowane są wszystkie przypadki uczące. Po zrealizowaniu epoki przeprowadza się zwykle ocenę sieci, przy pomocy zbioru walidacyjnego.

Wskaźnik tolerancji służy do określenia dopuszczalnego błędu na wyjściu sieci. Wskaźnik tolerancji jest związany ze wskaźnikiem ilości wzorców poza tolerancją. Też możemy ustawić wskaźnik, ile wzorców może znaleźć się poza tolerancją, a i tak akceptowany jest wynik.

W eksperymentach porównywano jakość uczenia, jakość testowania, jakość walidacji, a tym samym błąd uczenia, błąd testowania oraz błąd walidacji.

W sieciach klasyfikujących *jakość dla podzbioru uczącego, testowego i walidacyjnego* została obliczona jako proporcja poprawnie zaklasyfikowanych przypadków

w stosunku do wszystkich przypadków ze zbioru. Dla zespołu sieci jakość została obliczona jako ważona średnia jakości sieci składowych.

Błędem uczenia, testowania i walidacji w pierwszych eksperymentach był *błąd RMS*. Kolejno w eksperymentach jako funkcję błędu wykorzystano *entropię wzajemną (pojedynczą i wielokrotną)*. Błąd ten jest sumą iloczynów zadanych wartości oraz logarytmów błędów dla każdego neuronu wyjściowego. Funkcja ta występuje w dwóch wersjach: jedna z nich przeznaczona jest dla sieci z pojedynczym neuronem wyjściowym (rozróżniającym dwie klasy), druga zaś dla sieci z wieloma neuronami wyjściowymi. Funkcja błędu oparta na entropii wzajemnej została specjalnie zaprojektowana dla problemów klasyfikacyjnych, gdzie jest stosowana łącznie z występującą w warstwie wyjściowej sieci logistyczną funkcją aktywacji (gdy przewidywane jest pojedyncze wyjście sieci) lub funkcją aktywacji typu Softmax (gdy w sieci występuje większa liczba wyjść).

Sieci neuronowe zwracają na wyjściu wartości ciągłe, dlatego znakomicie nadają się do szacowania i przewidywania. Sieci takie mogą analizować jednocześnie wiele zmiennych. Możliwe jest stworzenie modelu, nawet gdy rozwiązanie ma dużą złożoność.

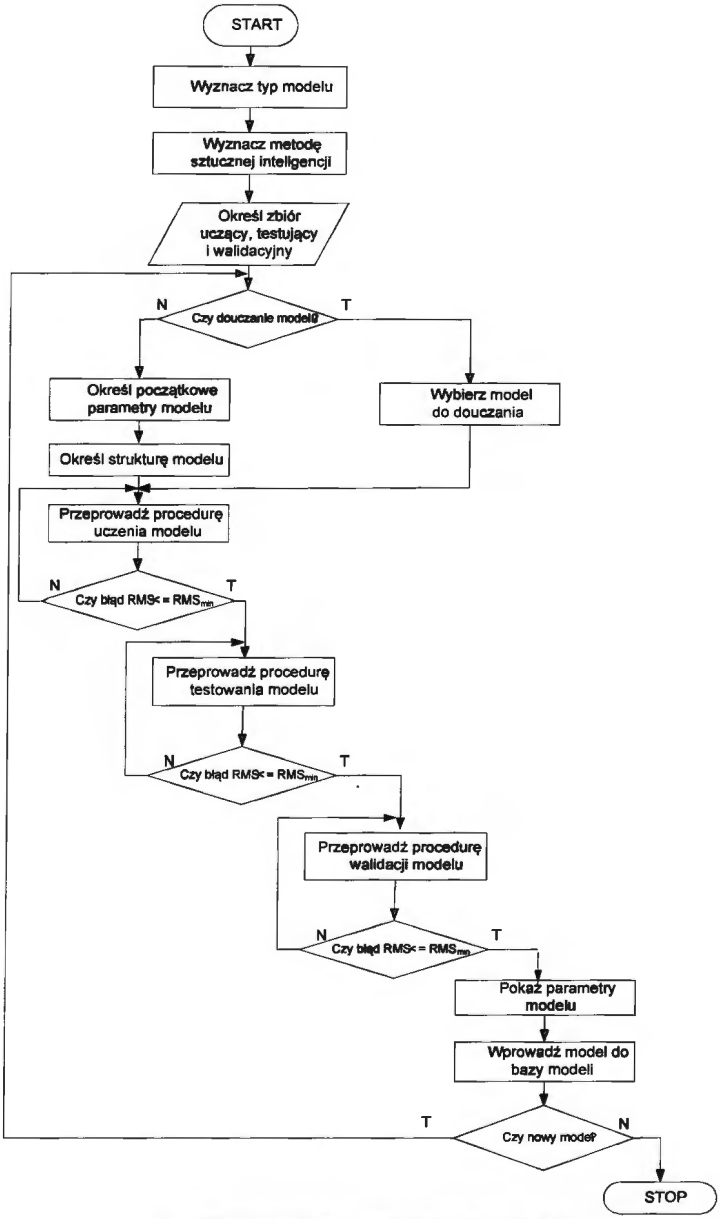
Wadami sieci neuronowych jest trudność w ustaleniu parametrów architektury, wpadanie w minima lokalne, długi czas uczenia się sieci i brak jasnej interpretacji.

Modele sieci neuronowej tworzone są według opracowanej wcześniej metodyki.

Na rysunku 10 pokazano algorytm tworzenia klasyfikatora. Algorytm składa się z następujących kroków [11]. Są one następujące:

1. Określ typ modelu, na przykład model klasyfikacji. Typ modelu oznacza, do czego ten model ma być wykorzystany, tutaj do klasyfikacji (sieci MLP i RBF), a inny model do grupowania (sieci Kohonena).
2. Wyznacz metodę sztucznej inteligencji: sieć neuronowa, zespół sieci neuronowych.
3. Określ zbiór uczący, testujący i walidacyjny.
2. Tworzenie nowego modelu:
    1. określ początkowe parametry modelu. Określa się wejścia i wyjście sieci, liczbę neuronów w warstwie ukrytej, funkcję błędu, funkcję aktywacji w warstwie ukrytej i warstwie wyjściowej, itp.
    2. określ strukturę modelu
    3. przeprowadź procedurę uczenia modelu
    4. czy zadane kryterium stopu (błąd RMS_{min}) zostało osiągnięte (błąd RMS <= błąd RMS_{min}), jeśli TAK, to przejdź do procedury testowania. Jeśli NIE to trwa procedura uczenia modelu
    5. przeprowadź procedurę testowania modelu
    6. czy zadane kryterium stopu zostało osiągnięte (błąd RMS <= błąd RMS_{min}), jeśli TAK, to pokaż parametry modelu. Jeśli NIE to trwa procedura testowania modelu
    7. przeprowadź procedurę walidacji modelu
    8. czy zadane kryterium stopu zostało osiągnięte (błąd RMS <= błąd RMS_{min}), jeśli TAK, to pokaż parametry modelu. Jeśli NIE to trwa procedura walidacji modelu
    9. wprowadź model do bazy modeli
    10. czy tworzysz nowy model? Jeśli TAK, to wróć do etapu: *Czy douczanie modelu?* Jeśli NIE, to STOP. Douczanie występuje wtedy, gdy pojawiają się nowe dane. Gotowy model bierzemy i douczamy na nowych danych. Jeśli nie ma, to nie robimy douczania.
4. W przypadku douczania modelu na nowych danych:

- wybierz model do douczania. Wybieramy gotowy model z bazy modeli, np. sieć neuronową, i douczamy nowym plikiem danych, który został uzupełniony o nowe dane.
- przeprowadź procedurę uczenia modelu
- czy zadane kryterium stopu zostało osiągnięte (błąd RMS <= błąd RMS_{min}), jeśli TAK, to przejdź do procedury testowania
- przeprowadź procedurę testowania modelu
- czy zadane kryterium stopu zostało osiągnięte (błąd RMS <= błąd RMS_{min}), jeśli TAK, to pokaż parametry modelu
- przeprowadź procedurę walidacji modelu
- czy zadane kryterium stopu zostało osiągnięte (błąd RMS <= błąd RMS_{min}), jeśli TAK, to pokaż parametry modelu. Jeśli NIE to trwa procedura walidacji modelu
- wprowadź model do bazy modeli
- czy tworzysz nowy model? Jeśli TAK, to wróć do etapu: *Czy douczanie modelu*? Jeśli NIE, to STOP.



Rys. 10. Algorytm tworzenia bazy modeli [11]