

Ontogeniczne sieci neuronowe

Norbert Jankowski & Włodzisław Duch

Katedra Metod Komputerowych
Uniwersytet Mikołaja Kopernika
ul. Grudziądzka 5, 87–100 Toruń
e-mail: Norbert.Jankowski@phys.uni.torun.pl
www: <http://www.phys.uni.torun.pl/~norbert>

Streszczenie

W pracy przedstawiono przegląd ontogenicznych modeli sieci neuronowych, czyli takich modeli, które dopasowują swoją strukturę (liczbę neuronów i połączeń pomiędzy nimi) do analizowanych danych. W szczególności dokładnie opisano model sieci IncNet, korzystający ze statystycznych kryteriów ocen optymalnej złożoności sieci.

1 Wstęp

Architektura sieci neuronowej, czyli układ warstw neuronów i połączeń pomiędzy neuronami, wraz z funkcjami transferu, określającymi sposób działania neuronów, determinuje złożoność i możliwości całego modelu adaptacyjnego¹.

Modele, które nie mogą modyfikować swojej architektury muszą ją mieć dobrze ustaloną na podstawie wiedzy *a priori* przed rozpoczęciem procesu uczenia. Jest to często bardzo trudne, ponieważ zazwyczaj nie jest znana złożoność rozwiązywanego problemu. Zakładając iż sieć neuronowa umie dobrze podejmować decyzje (możliwie najlepiej w każdej chwili uczenia) o zmianie swojej architektury, może ona dobrze kontrolować czy aktualna architektura modelu jest odpowiednia do reprezentowania rozwiązywanego problemu czy też jest zbyt lub za mało złożona.

Jeśli liczba powiązań lub neuronów danej sieci neuronowej będzie za mała model adaptacyjny nie będzie w stanie nauczyć się reprezentacji danego problemu, natomiast gdy połączeń między neuronami lub samych neuronów będzie za dużo, to model adaptacyjny nie będzie w stanie osiągnąć zadowalającego poziomu generalizacji (dojdzie do przeuczenia się sieci). Dlatego złożoność modelu adaptacyjnego powinna odpowiadać złożoności rozpatrywanego problemu. Wtedy model ma największą szansę na uzyskanie możliwie najlepszej jakości generalizacji.

Należy zwrócić uwagę, iż nie minimalna liczba neuronów, czy połączeń między neuronami powinna być głównym celem, lecz znalezienie takiej architektury, która umożliwi uzyskanie generalizacji jak najlepszej jakości. Aby sieć mogła ewoluować w kierunku uzyskania jak najlepszej generalizacji, musi posiadać pewien margines swobody, który leży w parametrach adaptacyjnych i pozwala zmieniać płynnie stany modelu. Dobrze dobrane marginesy swobody wraz z kryteriami kontroli złożoności modelu, pozwalają także walczyć z problemem lokalnych minimów. Model zmieniając swoją architekturę przechodzi do innych przestrzeni parametrów adaptacyjnych z inną funkcją błędów, w której następuje kontynuacja procesu uczenia, po czym znów mogą

¹Oczywiście abstrahując od algorytmu adaptacji sieci.

nastąpić zmiany architektury itd. Tym sposobem, taki model uczący może eksplorować bardzo różne przestrzenie w poszukiwaniu pewnego optimum. Dlatego należy zadbać, aby sam proces uczenia, jak i mechanizmy kontroli złożoności architektury mogły jak najefektywniej czerpać informacje ze zbioru treningowego i wszelkiej istniejącej wiedzy *a priori*. Na przykład, dodając do modelu informacje *a priori* o dokładności danych (tj. dokładności dokonanych pomiarów), na których opiera się proces adaptacji modelu, możemy osiągnąć lepsze wyniki.

W ostatnich latach powstało wiele modeli, które modyfikują swoją architekturę. Metody kontroli złożoności architektur sieci można podzielić na trzy grupy:

- powiększające — do tych modeli należą algorytmy, umożliwiające dokładanie nowych neuronów lub nowych połączeń pomiędzy neuronami;
- zmniejszające — metody usuwające zbędne neurony czy połączenia między neuronami lub algorytmy, które potrafią łączyć grupy neuronów lub połączenia pomiędzy neuronami;
- systemy kooperujące — grupy modeli, z których każdy indywidualnie ma za zadanie rozwiązywać ten sam problem lub jakąś jego część, a system zarządzający podejmuje ostateczną decyzję.

Najczęściej jednak spotyka się systemy, które należą do jednej z powyższych grup. Rzadziej spotyka się systemy, które należą do grupy systemów kooperujących, których podsystemy zmieniałyby swoją architekturę. Dla modeli ontogenicznych najważniejsze są kryteria, które decydują o tym, czy i/lub kiedy należy dokonać zmian w strukturze modelu, oraz której części struktury ma dotyczyć dana zmiana oraz jak jej dokonać.

Od skuteczności tych kryteriów uzależnione jest osiągnięcie dobrego poziomu generalizacji. Należy zwrócić uwagę by zmiany dokonywane na istniejącej strukturze nie doprowadzały do pogorszenia istniejącego poziomu *reprezentacji* problemu, lecz umożliwić jej polepszenie w dalszej procedurze adaptacji. Wybór części struktury, która ma podlegać zmianie też nie jest trywialny. Neuron lub pewne połączenie w danej chwili może nie odgrywać istotnej roli w sieci neuronowej gdyż proces adaptacji w intensywnej fazie uczenia mógł nie zdążyć go wykorzystać.

Niektóre systemy ontogeniczne mogą estymować rozkład danych opisujących problem zmienny w czasie. Jeśli złożoność samego problem podlega zmianie w czasie, wymagane są bieżące zmiany architektury sieci, tak, aby model mógł odzwierciedlać zmienioną złożoność problemu. Poza proponowaną przeze mnie siecią IncNet, opisywaną w tej pracy do estymacji niestacjonarnych rozkładów Fritzsche zaproponował inny model opisany w [24]. Aby dany model uczący mógł estymować rozkłady zmienne w czasie, musi kontrolować złożoność swojej aktualnej struktury. Większość spotykanych algorytmów dopuszcza jedynie usuwanie neuronów podczas uczenia albo ich dokładanie. W modelach rozrastających się neurony dokładane nierzadko są zamrażane i nie podlegają dalszej adaptacji. Można się spodziewać, że w przyszłości modele, które będą mogły szybko estymować zmienne rozkłady, będą wykorzystywane do symulacji zaawansowanych systemów kognitywnych.

2 Modele zmniejszające strukturę sieci

Modele, które potrafią zmniejszać swoją strukturę poprzez usuwanie połączeń lub usuwanie neuronów, próbują wykorzystywać informację zawartą w samych danych

i/lub w aktualnym stanie sieci (wartości wag, progów, rozmyć, etc.) i jej architekturze.

Najprostszy chyba sposób na usuwanie nieprzydatnych neuronów wyznacza współczynniki *istotności* lub *przydatności* każdego neuronu w bieżącej strukturze sieci:

$$s_i = E(\text{bez neuronu } i) - E(\text{z neuronem } i) \quad (1)$$

które wyznaczają różnice pomiędzy błędem sieci uzyskanym *bez* i *z* udziałem neuronu *i*. Sposób ten wymaga sporych nakładów obliczeniowych — do wyznaczenia każdego współczynnika s_i równania (1) należy wyznaczyć błąd dla całego zbioru treningowego. Neurony, dla których współczynniki istotności są znacznie mniejsze od wartości funkcji błędu można usunąć.

Zbliżony (również pasywny²) sposób wyznaczania współczynników istotności został użyty w rozwijanym w naszym zespole systemie FSM (*ang. Feature Space Mapping*) [1, 10, 13, 11] (por. rozdział 3). Współczynniki istotności wyznacza się dla każdego neuronu warstwy ukrytej po przerwaniu procesu uczenia w następujący sposób:

$$Q_i = C_i(\mathbf{X})/|\mathbf{X}| \quad (2)$$

gdzie $|\mathbf{X}|$ jest liczbą wzorców uczących, a $C_i(\mathbf{X})$ określa liczbę poprawnie sklasyfikowanych wzorców ze zbioru \mathbf{X} przez *i*-ty neuron. W sieci FSM każdy neuron warstwy ukrytej odpowiada pewnej podprzestrzeni (tj. klastrowi), z którą związana jest informacja o jego przynależności do pewnej klasy. Usuwane są te neurony, których współczynniki Q_i są bliskie zeru.

2.1 Modele zmniejszające strukturę a regularyzacja

Metody zmniejszające strukturę sieci neuronowej często można rozpatrywać jako metody regularyzacji. Wzorcowym przykładem może być rozpad wag (*ang. weight decay*) [30], gdzie do miary standardowej błędu modelu $E_0(f)$

$$E_0(f) = \frac{1}{2} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 \quad (3)$$

zostaje dodany czynnik regularyzacyjny:

$$E_{wd}(f, \mathbf{w}) = E_0(f) + \lambda \sum_{i=1}^M w_i^2 \quad (4)$$

Ustalając pewien próg θ , możemy dokonać usunięcia połączeń czy neuronów, których wartości wag są mniejsze od progu θ po zakończeniu lub przerwaniu procesu uczenia. Powyższa funkcja błędu wymusza jednak powstanie wielu małych wag, by uciec od powyższego problemu Weigend (m. in.) [49, 50] zaproponował eliminację wag (*ang. weight elimination*), stosując istotnie inny człon regularyzacyjny

$$E_{we}(f, \mathbf{w}) = E_0(f) + \lambda \sum_{i=1}^M \frac{w_i^2/w_0^2}{1 + w_i^2/w_0^2} \quad (5)$$

²Przez pasywność rozumie się tu brak związku z samym algorytmem uczenia, jak i samą funkcją błędu.

gdzie w_0 jest pewnym ustalonym współczynnikiem. Eksperymenty wykazały, iż współczynnik w_0 rzędu jedności odpowiada wartością aktywacji podobnego rzędu. Taki człon regularyzacyjny pozwala, aby istniała pewna liczba wag (większych od w_0) przyjmujących duże optymalne wartości. Waga w_i dla której $|w_i| \gg w_0$ ma wkład bliski wartości λ , natomiast dla wag w_i dla których $|w_i| \ll w_0$, wkład jest bliski zeru.

Parametr λ może podlegać adaptacji podczas uczenia. Adaptacje można przeprowadzać raz na epokę, zgodnie z poniższym schematem:

- $\lambda = \lambda + \Delta\lambda$ gdy $E_n < D \vee E_n < E_{n-1}$
- $\lambda = \lambda - \Delta\lambda$ gdy $E_n \geq E_{n-1} \wedge E_n < A_n \wedge E_n \geq D$
- $\lambda = 0.9\lambda$ gdy $E_n \geq E_{n-1} \wedge E_n \geq A_n \wedge E_n \geq D$

E_n jest błędem ostatniej (n -tej) epoki uczenia dla całego zbioru treningowego, $A_n = \gamma A_{n-1} + (1 - \gamma)E_n$ (γ jest bliskie 1), natomiast D określa błąd docelowy dla całego procesu uczenia.

Dodanie do standardowej funkcji błędu $E_0(f)$ (3) członu

$$E_{mlp2ln}(f, \mathbf{w}) = E_0(f) + \lambda \sum_{i=1}^M w_i^2 (w_i - 1)^2 (w_i + 1)^2 \quad (6)$$

wymusza zbieganie się wartości wag w pobliże 0 lub ± 1 . Człon ten został użyty do budowania i uczenia sieci MLP, służącej do ekstrakcji reguł logicznych [8].

Innym przykładem metody regresji, umożliwiającej kontrolę struktury sieci, może być lokalna regresja grzbietowa (*ang. local ridge regression*):

$$E_{lrr}(f, \mathbf{w}) = E_0(f) + \sum_{i=1}^M \lambda_i w_i^2 \quad (7)$$

W przypadku takiej regresji dla lokalnych funkcji, takich jak większość radialnych funkcji bazowych, gładkość regresji nie jest kontrolowana jednym współczynnikiem, lecz każda z funkcji jest kontrolowana niezależnie. To prowadzi do lokalnej adaptacji gładkości w zależności od stanu w lokalnej części przestrzeni. Regresja nielokalna dla problemów, w których gładkość funkcji w różnych częściach przestrzeni powinna być różna, często nie daje pożądanych rezultatów [46]. Do wyznaczania parametrów regularyzacyjnych można stosować uczenie poprzez krosvalidację (*ang. cross-validation*, co można przetłumaczyć jako *krzyżową wiarygodność*, ale nie ma powszechnie przyjętego terminu w języku polski) [46, 26] i różne jej odmiany.

Innymi metodami zmniejszającymi struktury sieci mogą być także metody współdzielenia wag poprzez różne neurony. Pośród tych metod ciekawa wydaje się metoda miękkiego współdzielenia wag (*ang. soft weight sharing*) [45], która nie wymaga, aby wagi danej grupy były sobie równe, lecz dopuszcza pewien rozrzut σ_j . Funkcja błędu z takim członem regularyzacyjnym przyjmuje postać

$$E_{sws}(f, \mathbf{w}) = E_0(f) - \lambda \sum_i \ln \left(\sum_{j=1}^k a_j \phi_j(w_i) \right) \quad (8)$$

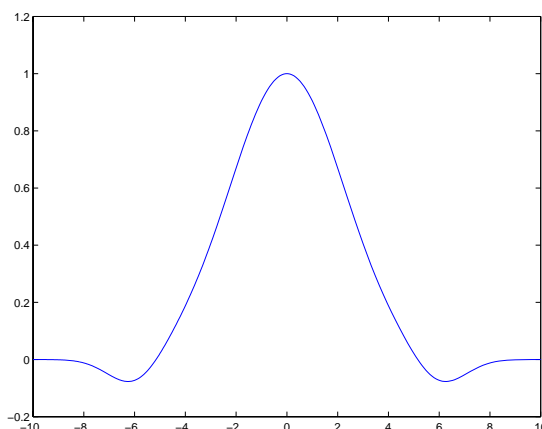
gdzie $\phi(w_i)$ jest zdefiniowane poprzez

$$\phi_j(w) = \frac{1}{(2\pi\sigma_j^2)^{1/2}} \exp\left(-\frac{(w - \mu_j)^2}{2\sigma_j^2}\right) \quad (9)$$

Taki człon również może pełnić rolę nie tylko czysto regularyzacyjną, ale i umożliwiać usuwanie połączeń. Parametry σ_j podlegają adaptacji, co może doprowadzić do stanu w którym pewne grupy wag będą odpowiadały bardzo podobnym wagom, a w takim przypadku można dokonać ich połączenia.

2.2 Usuwanie wag metodami *Optimal Brain Damage (OBD)* i *Optimal Brain Surgeon (OBS)*

Nie zawsze użycie metody rozpadu wag, jak i eliminacji wag daje wystarczająco dobre rezultaty. Czasem aby uzyskać naprawdę mały błąd, niezbędne okazują się i małe wagi. Dla przykładu popatrzmy na rysunek 1 funkcji o kształcie meksykańskiego kapelusza. Dobra aproksymacja wymaga użycia trzech funkcji gaussowskich, w tym dwóch o znacznie mniejszych wagach.



Rysunek 1: Meksykański kapelusz.

Dlatego też Le Cun i in. zaproponowali jeszcze inną metodę usuwania neuronów opartą o współczynniki istotności i nazwali ją *Optimal Brain Damage (OBD)* [7].

Droga rozwiązania problemu wiedzie przez analizę różnicy funkcji błędu δE , jaka powstaje pod wpływem zaburzenia wektora wag sieci o $\delta \mathbf{w}$. Analizuje się funkcję błędu, rozwiniętą w szereg Taylora

$$\delta E = \left(\frac{\partial E}{\partial \mathbf{w}}\right)^T \cdot \delta \mathbf{w} + \frac{1}{2} \delta \mathbf{w}^T \cdot \frac{\partial^2 E}{\partial \mathbf{w}^2} \cdot \delta \mathbf{w} + O(\|\delta \mathbf{w}\|^3) \quad (10)$$

gdzie $\frac{\partial^2 E}{\partial \mathbf{w}^2}$ tworzy Hessian H .

W minimum funkcji błędu pierwszy człon sumy prawej strony równania (10) można pominąć. Również i trzeci składnik dla uproszczenia można pominąć. Le Cun

zakłada również, iż tylko przekątna macierzy Hessianu jest rzeczywiście istotna, co prowadzi do uproszczenia równania (10) do postaci

$$\delta E = \frac{1}{2} \sum_i^M H_{ii} \delta w_i^2 \quad (11)$$

gdzie H_{ii} to i -ty element diagonalny macierzy Hessianu H . Z równania (11) widać, jak zmiana poszczególnych wag w_i może wpływać na zmiany funkcji błędu. Usunięcie i -tej wagi oznacza $\delta w_i = w_i$, co umożliwia określenie współczynników istotności s_i dla każdej wagi w_i

$$s_i = H_{ii} w_i^2 \quad (12)$$

Ostateczny algorytm składa się z następujących etapów:

1. Wstępny wybór architektury,
2. Uczenie sieci do etapu, w którym zmiany funkcji błędu nie będą już istotne,
3. Wyznaczenie współczynników istotności zgodnie z (12),
4. Usunięcie wag, których współczynnik istotności jest niski,
5. Jeśli któraś z wag została usunięta, to następuje skok do punktu 2.

Metoda ta została pomyślnie użyta do rozpoznawania pisma ręcznego, uzyskując błąd rzędu 3.4% na zbiorze testowym (2700 wektorów). Sieć była uczona na 9840 wektorach. Sama sieć składała się z pięciu wyspecjalizowanych warstw [6].

Następnym ciekawym krokiem była metoda zaprezentowana przez Hassibiego i Storka nazwana Optimal Brain Surgeon [27, 28]. Hassibi (i. in.) twierdzą, iż zazwyczaj macierz Hessianu wag jest zupełnie niediagonalna i prowadzi do usuwania dobrych, czyli istotnych wag. Pomijając pierwszą i trzecią część prawej strony równania (10) należy zminimalizować

$$\min_q \min_{\delta \mathbf{w}} \left\{ \frac{1}{2} \delta \mathbf{w}^T H \delta \mathbf{w} \right\} \quad \text{pod warunkiem } \delta w_q + w_q = 0 \quad (13)$$

Rozwiązanie za pomocą mnożników Lagrange'a

$$L = \frac{1}{2} \delta \mathbf{w}^T H \delta \mathbf{w} + \lambda (\delta w_q + w_q) \quad (14)$$

prowadzi do rozwiązania i wyznaczenia współczynników istotności dla wag s_q :

$$s_q = \frac{1}{2} \frac{w_q^2}{H_{qq}^{-1}} \quad (15)$$

$$\delta \mathbf{w} = -\frac{w_q}{H_{qq}^{-1}} H^{-1} \cdot \mathbf{i}_q \quad (16)$$

\mathbf{i}_q jest wektorem składającym się z zer i jedynki na q -tej pozycji. Z kolei $\delta \mathbf{w}$ to poprawki dla wag, jakie należy nanieść po usunięciu wagi w_q .

Proces uczenia sieci przebiega podobnie jak dla sieci z OBD (patrz powyżej). Hassibi prezentuje też zależność pomiędzy usuwaniem połączeń bazującym na wielkości wag, a metodami OBD i OBS

$$E(\text{wagi}) \geq E(\text{OBD}) \geq E(\text{OBS}) \quad (17)$$

Oba algorytmy OBD i OBS mogą być stosowane do sieci MLP, jak i do sieci RBF.

2.3 Statystyczne i inne metody zmniejszania struktury sieci neuronowych

Statystyczne podejście do usuwania zbędnych połączeń zostało przedstawione przez Finnoffa i Zimmermanna [19, 18] oraz Cottrella (i. in.) [5], a później użyte też w pracy Weigend'a (i. in.) w [51]. Idea tej metody opiera się na kumulowaniu różnic poszczególnych wag podczas uczenia w ramach jednej epoki. Następnie definiuje się współczynniki s_i , oceniające przydatność dla każdej wagi i . Współczynnik s_i jest równy ilorazowi wartości wagi powiększonej o średnie wahanie wartości wagi podzielonemu przez standardowe odchylenie średniej różnicy tej wagi:

$$s_i = \frac{|w_i + \overline{\Delta w_i^j}|}{std(\Delta w_i^j)} \quad (18)$$

w_i jest wartością i -tej wagi w poprzedniej epoce, Δw_i^j jest równe zmianie wartości wagi w_i pod wpływem prezentacji j -tego wzorca uczącego. Poza tym mamy

$$\overline{\Delta w_i^j} = \frac{1}{N} \sum_{j=1}^N \Delta w_i^j \quad (19)$$

$$std(\Delta w_i^j) = \sqrt{\frac{1}{N} \sum_{j=1}^N (\Delta w_i^j - \overline{\Delta w_i^j})^2} \quad (20)$$

Współczynnik s_i , testujący wagę i określony równaniem (18), jest duży gdy waga jest duża, a przeciętne różnice zmian tej wagi są małe. W odwrotnym przypadku mamy do czynienia z wagą, która jest raczej mało istotna lub zupełnie nieprzydatna.

W pracy [18] została opisana też procedura *epsi-prune*, której zadaniem nie jest pełne usunięcie zbędnej wagi, lecz jej dezaktywacja, która może być czasowa. Zanim dojdzie do usuwania wag sieć zostaje poddana procedurze uczenia, aż do przeuczenia się (np. aż do zaobserwowania pogorszenia błędu klasyfikacji bądź aproksymacji na podzbiorze zbioru treningowego wyodrębnionym do testowania). Następnie, po kilku epokach, wyznacza się współczynniki s_i zgodnie z równaniem (18). Wtedy następuje dezaktywacja tych wag, dla których współczynniki s_i są mniejsze od ϵ ($\epsilon > 0$). Natomiast wagi, które już zostały dezaktywowane poprzednio, a teraz ich współczynniki s_i są większe od ϵ , podlegają aktywacji z małą losową wartością początkową. Po za tym w każdej epoce wartość ϵ jest powiększana o wartość $\Delta\epsilon$ ($\Delta\epsilon > 0$), aż do osiągnięcia wartości ϵ_{max} .

Wartym wspomnienia jest również podejście Orr'a [46] do sieci RBF, bazujące na macierzy projekcji P , która wspomaga dokonywanie różnych operacji. Macierz projekcji jest zdefiniowana jako

$$P = I - HA^{-1}H^T \quad A = H^T H + \Lambda \quad (21)$$

gdzie Λ jest macierzą przekątniową, opisującą parametry regularyzacyjne poszczególnych wymiarów, a H jest macierzą wartości funkcji bazowych dla poszczególnych wektorów treningowych:

$$H = \begin{bmatrix} h_1(\mathbf{x}_1) & \dots & h_M(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_n) & \dots & h_M(\mathbf{x}_n) \end{bmatrix} \quad (22)$$

Macierz projekcji P pojawia się przy rozpisaniu błędu jako popełni sieć RBF dla wektorów treningowych

$$\hat{y} - F = \hat{y} - Hw = (I - HA^{-1}H^T)\hat{y} = P\hat{y} \quad (23)$$

Macierz projekcji pozwala na szybkie wyznaczenie sumy błędu kwadratowego:

$$E = (\hat{y} - F)^T(\hat{y} - F) = \hat{y}^T P^2 \hat{y} \quad (24)$$

Także i obliczanie funkcji kosztu może być szybkie:

$$C = (Hw - \hat{y})^T(Hw - \hat{y}) + w^T \Lambda w = \hat{y}^T P \hat{y} \quad (25)$$

Usunięcie zbędnej funkcji bazowej h_j pociąga za sobą następującą operację na macierzy projekcji (zakłada się iż funkcja h_j została przesunięta do ostatniej kolumny macierzy P_m):

$$P_{m-1} = P_m + \frac{P_m h_j h_j^T P_m}{\lambda_j - h_j^T P_m h_j} \quad (26)$$

Takie usunięcie kosztuje n operacji arytmetycznych (n to liczba wektorów uczących), natomiast pełne przetrenowanie sieci wymagałoby $M^3 + nM^2 + p^2m$ operacji (M liczba funkcji bazowych). W podobny sposób można dodać nową funkcję bazową (por. poniżej).

3 Modele o strukturach rozrastających się

Jednym z pierwszych modeli rozrastających się była sieć do klasyfikacji wzorców binarnych Mezarda i Nadala [44]. Sieć powstaje poprzez dokładanie nowych warstw o coraz mniejszej liczbie neuronów w kolejnych warstwach. Połączone są tylko neurony pomiędzy sąsiednimi warstwami. Każda z warstw musi spełniać następujący warunek *wierności* zbiorowi treningowemu: aby dwa wzorce uczące, należące do różnych klas, nie mogły być odwzorowane na taki sam wzorec aktywacji danej warstwy. W przeciwnym razie dochodzi do sytuacji, w których dla wzorców różnych klas powstają aktywacje odpowiadającym tym samym typom aktywacji danej warstwy, co uniemożliwiło by separację takich wzorców poprzez następną warstwę neuronów. Dopóki algorytm nie może zbudować *wiernej* warstwy, następuje dodawanie i uczenie kolejnych neuronów, aż dana warstwa spełni ten warunek. Uczenie odbywa się przy pomocy algorytmu kieszonkowego [25]. Do uczenia wykorzystuje się wektory wejściowe, dla których uzyskano takie same aktywacje w budowanej warstwie, podczas gdy należały one do różnych klas (czyli wektory, które sprawiają, że nie jest spełniony warunek wierności). Główna idea algorytmu kieszonkowego polega na przetrenowaniu podzbioru wag, które nie ulegałyby zmianom podczas prezentacji wielu wektorów treningowych [25]. Tak zdefiniowany algorytm gwarantuje zbieżność procesu uczenia.

Następny algorytm (*ang. upstart*) również służy do klasyfikacji wzorców binarnych i został zaproponowany przez Freana [20]. Ten algorytm dokłada neurony pomiędzy już istniejącymi neuronami a wejściami sieci. Algorytm upstart startuje ucząc

jeden neuron algorytmem kieszonkowym. Gdy nadal są błędy dokładane są dwa potomne neurony pomiędzy neuron, który *popęlnia błędy*, a odpowiadające mu wejścia. Wagi neuronów ustala się na takie wartości, aby pierwszy neuron odpowiadał wzorcom niesklasyfikowanym przez neuron-ojca niepoprawnie, a drugi sklasyfikowanym, ale błędnie. Po zamrożeniu wag dochodzących do neuronu-ojca następuje przetrenowanie neuronów potomnych. Następnie i te wagi są zamrażane, i jeśli klasyfikacja nie jest satysfakcjonująca, dodawany jest kolejny neuron, po czym proces jest powtarzany.

Algorytm korelacji kaskadowej (*ang. cascade-correlation network*) (CC) Fahlmana i Lebiere'a [15, 16] jest jednym z najsprawniejszych algorytmów uczenia sieci MLP, który umożliwia rozrastanie się struktury aż do uzyskania sieci o wystarczającej pojemności. Fahlman i Lebiere używali do adaptacji wag algorytmu szybkiej wstecznej propagacji, choć można rozważać użycie i innych algorytmów adaptacji.

Sieć kaskadowej korelacji rozpoczyna z jedną warstwą wag pomiędzy wejściem i wyjściem sieci. Następnie sieć jest uczona przez pewien czas, po czym następuje dołożenie nowego neuronu do warstwy ukrytej. Proces ten jest powtarzany aż do uzyskania zakładanego poziomu błędu. Na wejścia kolejno dodawanych neuronów składają się wszystkie wejścia sieci i wyjścia poprzednio dodanych neuronów, skąd bierze się słowo kaskada w nazwie sieci.

Drugi człon nazwy sieci związany jest z maksymalizacją poniższego wyrażenia (27) mierzącego korelację pomiędzy aktywacjami neuronów i popełnianymi błędami, którego celem jest ustalenie możliwie najlepszych wag dla nowego neuronu

$$S = \sum_{i=1}^p \left| \sum_{j=1}^n (o_j - \bar{o})(e_j^i - \bar{e}^i) \right| \quad (27)$$

gdzie n oznacza liczbę wzorców uczących, p jest liczbą wyjść sieci, o_j jest wartością aktywacji neuronu, który ma być dodany dla j -tego wzorca uczącego, \bar{o} jest średnią aktywacją dodawanego neuronu, e_j^i jest błędem i -tego wyjścia dla j -tego wzorca uczącego, a \bar{e}^i jest średnim błędem i -tego wyjścia. Łatwo można wyznaczyć pochodną równania (27)

$$\frac{\partial S}{\partial w_k} = z_i \sum_{i=1}^p \sum_{j=1}^n (e_j^i - \bar{e}^i) g_j' I_j^k \quad (28)$$

z_i jest równe 1 lub -1 zależnie od tego, czy korelacja pomiędzy neuronem i i -tym wyjściem sieci jest dodatnia lub ujemna; g_j' jest wartością pochodnej funkcji transferu dodawanego neuronu dla j -tego wektora uczącego, z kolei I_j^k jest wartością k -tego wejścia dodawanego neuronu dla j -tego wektora uczącego.

Start uczenia neuronu, który ma zostać dodany, rozpoczyna się od wartości losowych, dlatego też często wybiera się kilka neuronów-kandydatów, z których następnie wybiera się lepszego po wstępnym procesie adaptacji, w którym maksymalizuje się korelację. Fahlman opracował również rekurencyjną wersję wyżej opisanego algorytmu [14].

Innym, ciekawym modelem, umożliwiającym rozbudowywanie struktury podczas uczenia się przez dodawanie nowych neuronów, jest wspomniany już system FSM, rozwijany w naszym zespole [1, 10, 13]. Model ten był używany głównie do klasyfikacji i wyciągania reguł logicznych z danych.

Architektura FSM jest praktycznie taka sama, jak sieci RBF. Składa się z warstwy wejściowej, wyjściowej i jednej warstwy ukrytej. Na wyjściu pojawia się informacja o tym, do której klasy został przypisany wektor pokazany warstwie wejściowej. Jednakże najważniejszą część stanowi warstwa ukryta, która jest odpowiedzialna za konstrukcje odpowiednich zbiorów odwzorowań. Mamy więc nie tylko podobieństwo architektur sieci RBF, ale i roli ich warstw ukrytych. W zastosowaniach klasyfikacyjnych, jak i w ekstrakcji reguł logicznych, neurony warstwy ukrytej odpowiadają pewnej (zależnej od funkcji transferu) klasteryzacji przestrzeni wejściowej. Typowymi funkcjami transferu neuronów warstwy ukrytej są funkcje gaussowskie wielu zmiennych, funkcje bicentralne [12, 35], jak i funkcje definiujące hiperprostopadłościany.

Inicjalizacja architektury jest dokonywana za pomocą algorytmów klasteryzacji: poprzez histogramy, drzewa decyzyjne lub dendrogramy [9]. Po procesie inicjalizacji następuje etap estymacji położenia, rozmycia i innych parametrów algorytmem opisanym w [1, 10]. Główna część procesu estymacji oparta jest o analizę poszczególnych wektorów uczących dla tego neuronu, dla którego uzyskano największą aktywację w wyniku prezentacji danego wektora uczącego i neuronu najbliższego neuronowi, dla którego uzyskano największą aktywację. Algorytm adaptacji umożliwia dodanie nowego neuronu, gdy są spełnione trzy warunki:

1. dla danego wektora uczącego \mathbf{x} , neuron, dla którego została uzyskana największa aktywacja N_M i neuron jemu najbliższy N_N , należą do różnych klas,
2. odległość, pomiędzy wektorem uczącym, a neuronem maksymalnie pobudzonym N_M , spełnia poniższą nierówność

$$\|\mathbf{x} - t_{N_M}\| > \sigma_{N_M} \sqrt{\ln 10} \quad (29)$$

3. maksymalna aktywacja, uzyskana dla neuronu N_M , jest mniejsza niż ustalona wartość Act_{min}

$$G_{N_M}(\mathbf{x}) < Act_{min} \quad (30)$$

Wspomniana już sieć MLP2LR [8], służąca do wyciągania reguł logicznych, również umożliwia dodawanie nowych neuronów, a samo kryterium niewystarczalności struktury sieci jest zdefiniowane w bardzo prosty sposób: sieć przestała się uczyć (wartość funkcji błędu przestała maleć) i poprawność klasyfikacji jest wciąż zbyt mała.

W podrozdziale 2.3 opisano m. in. metodę usuwania funkcji bazowych, opisaną przez Orr'a [46] za pomocą operacji na macierzy projekcji P . Wykonując inne, choć podobne, operacje na macierzy projekcji, można dodać nowy neuron. Odpowiednie zmiany w macierzy projekcji P_m pokazane są poniżej

$$P_{m+1} = P_m - \frac{P_m h_{m+1} h_{m+1}^T P_m}{\lambda_j + h_{m+1}^T P_m h_{m+1}} \quad (31)$$

W pracach [21, 22] Fritzsche prezentuje sieć RBF, w której co λ kroków uczenia (prezentacji pojedynczego wektora uczącego) następuje dodanie nowego neuronu. Zanim to nastąpi podczas procesu adaptacyjnego wyznaczane są skumulowane błędy, jakie sieć popełnia w poszczególnych podobszarach, które reprezentują neurony

warstwy ukrytej. Przy każdej prezentacji kolejnego wektora uczącego następuje kumulowanie błędu, doliczając sumaryczny błąd kwadratowy neuronowi s , który był najbliższym prezentowanemu wektorowi uczącego:

$$\Delta err_s = \sum_{i=1}^d (F(\mathbf{x}^i) - y^i)^2 \quad (32)$$

$F(\mathbf{x})^i$ oznacza wartość i -tego wyjścia dla wektora \mathbf{x} , a y^i wartość oczekiwaną na i -tym wyjściu. Po upływie każdego λ kroków uczenia następuje dodanie nowego neuronu w pobliżu neuronu, dla którego aktualny skumulowany błąd jest największy. Dokładniej pozycja nowego neuronu jest wyznaczona jako średnia pozycji neuronu, dla którego skumulowany błąd był największy i jednego z jego najbliższych sąsiadów.

Informacje o innych modelach sieci samoorganizujących się można znaleźć w [24, 23]. W pracy [24] rozkład topologii sieci może podążać za zmieniającym się w czasie rozkładem danych.

Fiesler [17] dokonał porównania ponad 30 modeli ontogenicznych. Zestawienie zawiera informacje o liczbie warstw, czy sieć umożliwia rozrastanie się, czy sieć może się kurczyć, czy metoda jest lokalna, jakie są dodatkowe warunki nakładane na sieć (np. czy startuje z pustej struktury), czy istnieją jakieś matematyczne dowody na zbieżność metody. Jednakże zestawienie to nie wskazuje na jakiegokolwiek wady, czy zalety poszczególnych modeli.

3.1 Sieć RAN z przydziałem zasobów

Sieć z przydziałem zasobów (*ang. Resource Allocation Network, RAN*), została zaproponowana przez Platt w [47]. Nieco później Kadirkamanathan i Niranjana [42, 38] pokazali, iż, przy pewnych założeniach, użyty w sieci RAN sekwencyjny sposób uczenia jest poprawny matematycznie. Podobnie pokazano, że kryteria rozrostu sieci (nazwane geometrycznym kryterium rozrostu) są również poprawne przy założeniach, które nie stoją w opozycji do opisanego przez Platta modelu.

Sieć RAN można rozpatrywać, jako sekwencyjną metodę estymacji pewnego nieznanego gładkiego odwzorowania F^* , co oznacza, że celem jest wyznaczenie nowego stanu modelu $F^{(n)}$ na podstawie poprzedniego stanu modelu $F^{(n-1)}$ i nowej obserwacji $I^{(n)}$, która jest parą uczącą $\langle \mathbf{x}_n, y_n \rangle$ ($\mathbf{x} \in \mathcal{R}^N, y \in \mathcal{R}$).

Tak określone zadanie można zapisać w postaci funkcji celu \mathcal{F} -projekcji:

$$F^{(n)} = \arg \min_{f \in \mathcal{H}} \|f - F^{(n-1)}\| \quad F^{(n)} \in \mathcal{H}_n \quad (33)$$

gdzie \mathcal{H} jest przestrzenią Hilberta funkcji o skończonej normie w L^2 , a \mathcal{H}_n jest zbiorem funkcji określonych poprzez:

$$\mathcal{H}_n = \{f : f \in \mathcal{H} \wedge f(\mathbf{x}_n) = y_n\} \quad (34)$$

Warunek $F(\mathbf{x}_n) = y_n$ można zapisać jako iloczyn skalarny funkcji należących do przestrzeni \mathcal{H} :

$$\langle F, g_n \rangle = \int_{\mathbf{x} \in \mathcal{D}} F(\mathbf{x})g(\mathbf{x} - \mathbf{x}_n) = y_n \quad (35)$$

gdzie $g(\cdot)$ jest funkcją impulsową. To pozwala na zapisanie funkcji celu jako \mathcal{F} -projekcji w postaci modelu *a posteriori*:

$$F^{(n)} = F^{(n-1)} + e_n \frac{g_n}{\|g_n\|^2} = F^{(n-1)} + e_n h_n \quad (36)$$

e_n jest błędem popełnionym przez model *a priori* $F^{(n-1)}$ w punkcie \mathbf{x}_n ($e_n = y_n - F^{(n-1)}(\mathbf{x}_n)$).

Takie rozwiązanie wprowadza bardzo ostrą funkcję impulsową w punkcie \mathbf{x}_n do już istniejącego modelu $F^{(n-1)}$ tak, aby model w kolejnym stanie osiągał wartość y_n w punkcie \mathbf{x}_n . To prowadzi do ograniczenia na gładkość funkcji $h_n(\cdot)$, która może być użyta w naszym rozwiązaniu (36)

$$h_n(\mathbf{x}_n) = 1 \quad \wedge \quad h_n(\mathbf{x}_n + a) = 0 \quad \text{dla } \|a\| > 0 \quad (37)$$

Dobrym przykładem takiej funkcji $h_n(\cdot)$ może być funkcja gaussowska z odpowiednio dobranym współczynnikiem rozmycia b

$$G(\mathbf{x}; \mathbf{t}, b) = e^{-\|\mathbf{x}-\mathbf{t}\|^2/b^2} \quad (38)$$

Podsumowując, korzystając z \mathcal{F} -projekcji, powyższego założenia o gładkości i że poprzedni model $F^{(n-1)}$ składał się z M funkcji bazowych (neuronów) mamy:

$$F^{(n)} = \sum_{i=1}^M w_i G(\mathbf{x}; \mathbf{t}_i, b_i) + e_n G(\mathbf{x}; \mathbf{x}_n, b_0) \quad (39)$$

$$= \sum_{i=1}^{M+1} w_i G(\mathbf{x}; \mathbf{t}_i, b_i) \quad (40)$$

z $\mathbf{t}_{M+1} = \mathbf{x}_n$, $b_{M+1} = b_0$, b_0 jest początkowym rozmyciem. Oznacza to rozbudowującą się sieć RBF, której punktem wyjścia jest pusta warstwa ukryta.

Z powyższych rozważań widać, iż sekwencyjny ciąg kolejnych modeli, wybierając odpowiednio małą wartość początkową b_0 , może w rezultacie prowadzić do dowolnie małego ustalonego błędu interpolacji ϵ .

Geometryczne kryterium rozrostu. Oczywiście każdy kolejny stan k modelu $F^{(n+k)}$ nie powinien kosztować dodania nowej funkcji bazowej. Dlatego też sieć RAN wykorzystuje geometryczne kryterium rozrostu, umożliwiające kontrolę złożoności sieci, w zależności od aktualnego jej stanu i napływających nowych obserwacji I_n podczas procesu sekwencyjnego uczenia.

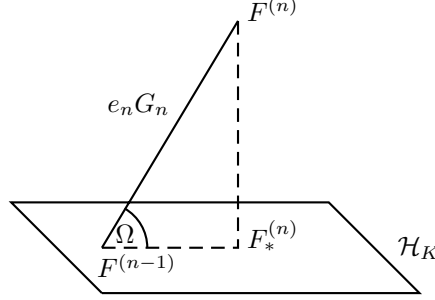
Aby odpowiedzieć na powyższe pytanie, czy model $F^{(n)}$ wystarczy, aby dostatecznie dobrze estymować nową obserwację I_n , trzeba prześledzić sytuację, w której model nie zostaje powiększony o nowy neuron i sytuację, w której model zostaje rozbudowany o nową funkcję bazową (patrz rysunek 2). Modele *a priori* $F^{(n-1)}$ i *a posteriori* $F_*^{(n)}$ (nie powiększony o nową funkcję bazową) należą do przestrzeni \mathcal{H}_M . Natomiast model *a posteriori* $F^{(n)}$ powiększony o nową funkcję bazową należy już do przestrzeni \mathcal{H}_{M+1} . Model $F_*^{(n)}$ jest projekcją modelu $F^{(n)}$ do przestrzeni \mathcal{H}_M .

Jeśli odległości $\|F^{(n)} - F_*^{(n)}\|$ jest większa od pewnego ϵ () to przestrzeń funkcji \mathcal{H}_M jest niewystarczająca do uzyskania akceptowalnego poziomu estymacji:

$$\|F^{(n)} - F_*^{(n)}\| > \epsilon \quad (41)$$

Poza tym mamy też następującą własność (porównaj z rys. 2):

$$\|F^{(n)} - F_*^{(n)}\| = |e_n| \cdot \|G_n\| \sin(\Omega) \quad (42)$$



Rysunek 2: Zależności pomiędzy modelami a posteriori $F_*^{(n)}$ i $F^{(n)}$ (odpowiednio z przestrzeni \mathcal{H}_M i \mathcal{H}_{M+1}) względem modelu *a priori* $F^{(n-1)}$.

Ponieważ $\|G_n\|$ zależy tylko od stałej początkowego rozmycia b_0 , a z kolei kąt Ω może przyjmować wartości od 0 do $\pi/2$, można uprościć nierówność (41) do poniższych kryteriów geometrycznych:

$$e_n > e_{min} \quad (43)$$

$$\Omega > \Omega_{min} \quad (44)$$

Pierwsza z powyższych nierówności to *kryterium predykcji błędu*. Ocenia ono błąd interpolacyjny. Druga z powyższych nierówności to *kryterium kąta*, które ocenia na ile funkcja bazowa G_n jest ortogonalna (duży kąt) do funkcji bazowych z przestrzeni funkcji \mathcal{H}_M . W ogólnym przypadku wyznaczenie poszczególnych kątów jest trudne, ale można dokonać pewnych przybliżeń, na przykład przyjmując równe rozmycia funkcji G_n i funkcji G_i . Wtedy kąt pomiędzy funkcjami G_n i G_i jest równy:

$$\Omega_i = \cos^{-1} \left(\exp \left(-\frac{1}{2b_0^2} \|\mathbf{x}_n - \mathbf{t}_i\|^2 \right) \right) \quad (45)$$

Korzystając z powyższej własności można przepisać kryterium kąta (44) do postaci:

$$\sup_i G_i(\mathbf{x}_n) \leq \cos^2(\Omega_{min}) \quad (46)$$

co ze względu na monotoniczne nachodzenie się funkcji gaussowskiej przy oddalaniu się punktu \mathbf{x}_n od centrum funkcji G_i , można zastąpić przez:

$$\inf_i \|\mathbf{x}_n - \mathbf{t}_i\| \geq \epsilon \quad (47)$$

Powyższe kryterium przeradza się w *kryterium odległości* i tak naprawdę pokazuje, że porównaniu podlega odległość pomiędzy punktem \mathbf{x}_n i centrum najbliższej funkcji bazowej, a wartości ϵ

$$\epsilon = b_0 \sqrt{2 \log(1 / \cos^2 \Omega_{min})} \quad (48)$$

Można też określić optymalne rozmycie dla nowej funkcji bazowej, które będzie maksymalnie duże, ale będzie też spełniać kryterium kąta (44):

$$b_n = \frac{\|\mathbf{x}_n - \mathbf{t}_k\|}{\sqrt{2 \log(1/\cos^2 \Omega_{min})}} \quad k = \arg \min_k \|\mathbf{x}_n - \mathbf{t}_k\| \quad (49)$$

Podsumowując, stwierdzić można, że nowy neuron jest dodawany, gdy spełnione jest kryterium predykcji błędu (43) i kryterium odległości (47).

Adaptacja sieci RAN. Gdy nie jest spełnione kryterium predykcji błędu (43) lub kryterium odległości (47), następuje adaptacja wolnych parametrów sieci, wagi i położenia centrów funkcji bazowych ($\mathbf{p}^{(n)} = [w_0, w_1, \dots, w_M, \mathbf{t}_1^T, \mathbf{t}_2^T, \dots, \mathbf{t}_M^T]^T$) algorytmem LMS:

$$\mathbf{p}^{(n)} = \mathbf{p}^{(n-1)} + \eta e_n \mathbf{d}_n \quad (50)$$

gdzie \mathbf{d}_n jest gradientem funkcji $F(\mathbf{x}_n)$ po parametrach \mathbf{p} . Zastępując η przez $\frac{\eta}{\|\mathbf{x}_n\|^2}$, uzyskuje się znormalizowaną wersję algorytmu LMS.

W praktyce również odległość minimalna ϵ podlega zmianom podczas procesu uczenia. Początkowa wartość ϵ jest równa ϵ_{max} , a następnie podlega iteracyjnym zmianom $\epsilon = \epsilon_{max} \gamma^n$, dla $0 < \gamma < 1$, aż do osiągnięcia wartości ϵ_{min} .

4 Sieć IncNet ze statystyczną kontrolą złożoności sieci

Praktycznie wszystkie przedstawione powyżej modele ontogeniczne zawsze nakładają dość drastyczne uproszczenia dla całości procesu uczenia. Wynikają one z koncentracji uwagi nad częścią rozwiązywanego problemu, a nie na całości. Dla przykładu algorytmy OBD [7, 6], OBS[27, 28] czy FSM [1, 10, 13, 11], zezwalają na usuwanie neuronów praktycznie po zakończeniu właściwego procesu uczenia. Modele dodające człon regularyzacyjny wymuszają niskie wartości wag, co często prowadzi do zbyt małych wartości niemal wszystkich wag. Inne metody regularyzacji wymagają dobrego dobrania parametru (lub parametrów) na podstawie wiedzy a priori, co nie zawsze jest proste w praktyce. Rozważania przedstawione poniżej pokażą, iż można zdefiniować inne kryteria, które umożliwią usuwanie neuronów praktycznie z iteracji na iterację, czyli gdy tylko okaże się to korzystne dla procesu adaptacji, a nie co epokę lub po zakończeniu procesu uczenia, tak jak jest to realizowane w innych algorytmach.

Algorytmy, które mogą modyfikować swoją architekturę podczas uczenia, nie czynią tego w optymalny sposób. W sieci RAN spełnienie kryteriów (kryterium predykcji błędu i kryterium odległości) opisanych równaniami (43 i 47), nie zawsze odpowiada sytuacjom, w których sieć nie jest zdolna do uwzględnienia nowej obserwacji. Może to odpowiadać zbyt szybkiej prezentacji danej w dość odległym rejonie, do którego i tak jakaś z funkcji bazowych zostałaby *przyciągnięta* w miarę postępowania procesu uczenia. Spotyka się również naiwną taktykę dokładania nowych funkcji bazowych do sieci RBF co stałą liczbę kroków uczenia. Jeszcze inną wadą już nie tylko powyżej opisanych systemów ontogenicznych lecz znacznej części sieci neuronowych jest lekceważenie istotności wyboru funkcji transferu, które silnie determinują możliwości generalizacji modeli adaptacyjnych.

Głównym celem systemu IncNet, zaprezentowanego w kolejnych podrozdziałach, stało się stworzenie takiego modelu, który będzie korzystał z efektywnego algorytmu uczenia i jednocześnie będzie zdolny do auto-kontroli złożoności architektury sieci podczas procesu uczenia, tak aby złożoność architektury sieci odzwierciedlała złożoność zaprezentowanej dotychczas części zbioru wektorów uczących. Oczywiście tak postawione zadanie jest bardzo trudnym problemem i nie jest możliwe jego rozwiązanie w 100%. Proponowanym rozwiązaniem problemu może być system, który będzie łączył w sobie poniższe cechy:

Uczenie: wykorzystanie efektywnego filtra Kalmana do adaptacji swobodnych parametrów sieci,

Kontrola złożoności sieci: wykorzystanie kryteriów kontrolujących rozbudowywanie się sieci, jak i kurczenie się sieci, poprzez dodawanie, usuwanie i łączenie się neuronów,

Elastyczne funkcje transferu: użycie bicentralnych funkcji transferu umożliwiające znacznie efektywniejszą estymację, a w efekcie umożliwia adaptację dla złożonych problemów.

System, posiadający takie cechy, jest zdolny do efektywnego uczenia sieci neuro- nowej przy jednoczesnej kontroli użyteczności każdego z podelementów i złożoności całego systemu do estymacji nowych obserwacji.

Struktura sieci i funkcje transferu. Struktura sieci IncNet jest praktycznie taka sama jak sieci RBF (czy też RAN):

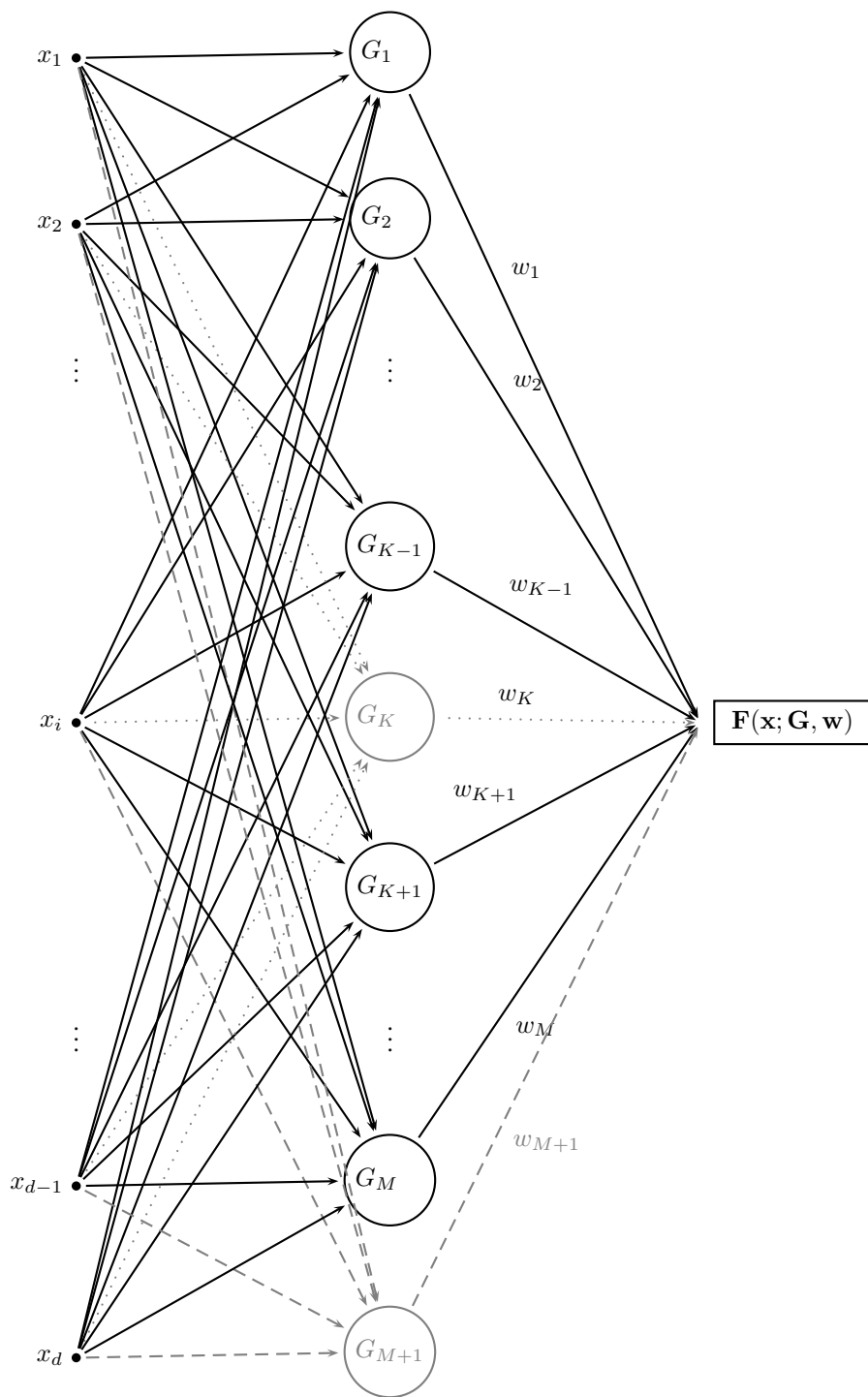
$$f(\mathbf{x}; \mathbf{w}, \mathbf{p}) = \sum_{i=1}^M w_i G_i(\mathbf{x}, \mathbf{p}_i) \quad (51)$$

Różni je jednakże fakt, iż struktura sieci RBF jest statyczna i nie podlega zmianom podczas uczenia, natomiast sieć IncNet jest dynamiczna i używa zmiennej liczby funkcji bazowych.

W standardowej sieci RBF, w sieci RAN i w pierwszej wersji sieci IncNet [39], jako funkcje bazowe używane były funkcje gaussowskie. Znaczny postęp uzyskano stosując w miejsce funkcji gaussowskich funkcje bicentralne, które powstają z produktu par funkcji sigmoidalnych:

$$Bi(\mathbf{x}; \mathbf{t}, \mathbf{b}, \mathbf{s}) = \prod_{i=1}^N \sigma(e^{s_i} \cdot (x_i - t_i + e^{b_i})) (1 - \sigma(e^{s_i} \cdot (x_i - t_i - e^{b_i}))) \quad (52)$$

Pozwalają one na estymację bardziej zróżnicowanych powierzchni decyzyjnych przy użyciu mniejszej liczby funkcji bazowych. Standardowa funkcja bicentralna umożliwia niezależną kontrolę rozmycia jak i skosu w każdym wymiarze niezależnie. Zaproponowane różne rozszerzenia funkcji bicentralnych umożliwiają rotację granic decyzji w wielowymiarowej przestrzeni, czy uzyskiwanie semi-lokalnych granic, jak i desymetryzacji w podwymiarach. Szczegółowo funkcje bicentralne zostały przedstawione w [12, 35].



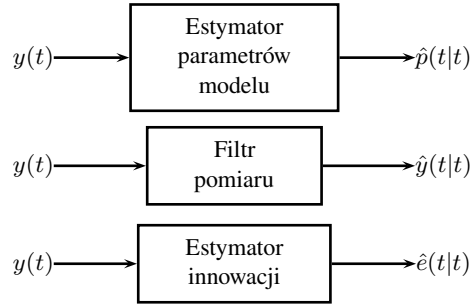
Rysunek 3: Struktura sieci IncNet. Sieć umożliwia dodawanie nowej funkcji bazowej G_{M+1} lub usuwanie pewnej funkcji bazowej G_K , jak również łączenie neuronów.

Rozszerzony filtr Kalmana. Definicja błędu dla uczenia sekwencyjnego sieci RAN przez \mathcal{F} -projekcję, zdefiniowaną równaniem (33), może zostać uogólniona do minimalizacji funkcji:

$$E_{sq} = \int_{\mathcal{D}} |F^{(n)}(\mathbf{x}) - F^{(n-1)}(\mathbf{x})|^2 p^{(n-1)} d\mathbf{x} + \frac{1}{n-1} |y_n - F^{(n)}(\mathbf{x}_n)|^2 \quad (53)$$

gdzie $n-1$ i n odpowiadają określeniu kolejnych stanów modelu adaptacyjnego, $p^{(n-1)}$ jest rozkładem prawdopodobieństwa ostatnich $n-1$ obserwacji (wektorów wejściowych). Powyższa funkcja błędu została użyta do algorytmu RNLS (*ang. recursive nonlinear least squares*) w pracach Kadirkamanathana i Niranjana [38, 40]. Minimalizacja powyższej funkcji błędu może być aproksymowana przez użycie rozszerzonej wersji algorytmu filtru Kalmana (EKF) [4, 29, 48, 43].

Filtr Kalmana jest estymatorem trzech różnych wyjść dla danej obserwacji (wektora danych), co do której zakłada się, że jest obciążona pewnym szumem, o zerowej wartości oczekiwanej, z pewnym niezerowym odchyleniem standardowym. Pierwsze z wyjść stanowi estymator parametrów modelu. To właśnie główny element uczenia sieci. Kolejne dwa wyjścia to filtr pomiaru i estymator innowacji (lub błędu). Wyjścia te będą wykorzystywane do estymacji i do kontroli złożoności całości modelu.



Filtr EKF prowadzi estymację parametrów *a posteriori* modelu \mathbf{p}_n , w oparciu o ich poprzedni stan *a priori* \mathbf{p}_{n-1} oraz błąd modelu e_n :

$$e_n = y_n - f(\mathbf{x}_n; \mathbf{p}_{n-1}) \quad (54)$$

i wartości wektora wzmocnienia Kalmana (*ang. Kalman gain*), które są pochodną macierzy kowariancji błędu *a priori* \mathbf{P}_{n-1} :

$$\mathbf{p}_n = \mathbf{p}_{n-1} + e_n \mathbf{k}_n \quad (55)$$

Wektor Kalmana \mathbf{k}_n jest wyznaczany przez:

$$\mathbf{k}_n = \mathbf{P}_{n-1} \mathbf{d}_n / R_y \quad (56)$$

\mathbf{d}_n jest wektorem gradientu funkcji modelu względem parametrów modelu:

$$\mathbf{d}_n = \frac{\partial f(\mathbf{x}_n; \mathbf{p}_{n-1})}{\partial \mathbf{p}_{n-1}} \quad (57)$$

natomiast R_y jest całkowitą wariancją modelu wyznaczaną poprzez:

$$R_y = R_n + \mathbf{d}_n^T \mathbf{P}_{n-1} \mathbf{d}_n \quad (58)$$

z kolei R_n określa wariancję szumu pomiarów, kontroluje proces regularyzacji. Estymacja a priori macierzy kowariancji błędu przebiega zgodnie z równaniem:

$$\mathbf{P}_n = [\mathbf{I} - \mathbf{k}_n \mathbf{d}_n^T] \mathbf{P}_{n-1} + Q_0(n) \mathbf{I} \quad (59)$$

\mathbf{I} jest macierzą jednostkową. Człon $Q_0(n) \mathbf{I}$ spełnia rolę (drobnego) losowego skoku w kierunku gradientu, wprowadzając małą perturbację w procesie adaptacji parametrów modelu i zapobiegając zbyt szybkiej zbieżności. Czasami umożliwia to *ucieczkę* z lokalnych minimów. $Q_0(n)$ może być bardzo małym dodatnim skalarzem bądź funkcją monotonicznie malejącą, przyjmującą bardzo małe dodatnie wartości (stopniowe zastyganie). Dla przykładu:

$$Q_0(n) = \begin{cases} Q_0 & n = 1 \\ Q_0(n-1) \cdot Q_{des} & n > 1 \wedge Q_0(n-1) \cdot Q_{des} > Q_{min} \\ Q_{min} & n > 1 \wedge Q_0(n-1) \cdot Q_{des} \leq Q_{min} \end{cases} \quad (60)$$

gdzie Q_0 jest wartością początkową dla $Q_0(n)$, Q_{des} definiuje szybkość zmniejszania pobudzania ($Q_{des} > 1$, zazwyczaj ok. 0.9988), a Q_{min} określa minimalną wartość $Q_0(n)$.

Wielką różnicę pomiędzy algorytmem EKF i LMS można zauważyć porównując równania (55) i (50) adaptacji parametrów p . W miejscu członu $\eta \mathbf{d}_n$ z algorytmu LMS, w filtrze EKF znajduje się wektor wzmocnienia Kalmana \mathbf{k}_n z równania (56), który wyznacza szybkość adaptacji każdego z parametrów nie tylko w zależności od wektora gradientu \mathbf{d}_n (jak to jest w algorytmie LMS czy stochastycznym spadku gradientu), lecz również w oparciu o macierz kowariancji \mathbf{P}_{n-1} . Właśnie to prowadzi do znacznie efektywniejszego procesu uczenia, radykalnie zmniejszając liczbę iteracji potrzebną do uzyskania zbieżności.

Szybka wersja rozszerzonego filtru Kalmana. Macierz kowariancji \mathbf{P}_n , w miarę przybywania nowych neuronów, może urosnąć do sporych rozmiarów (liczba elementów macierzy \mathbf{P}_n to kwadrat liczby parametrów adaptacyjnych). Estymacja tej macierzy może okazać się zbyt kosztowna. Rozsądną decyzją okazało się zredukowanie sporej części macierzy kowariancji przyjmując, że korelacje pomiędzy parametrami różnych neuronów nie są tak istotne, jak korelacje pomiędzy parametrami tego samego neuronu. Takie uproszczenie prowadzi do macierzy $\tilde{\mathbf{P}}_n$, która składa się z diagonalnego łańcucha podmacierzy $\tilde{\mathbf{P}}_n^k$ (elementy poza diagonalnym łańcuchem są równe zeru):

$$\tilde{\mathbf{P}}_n = \begin{bmatrix} \tilde{\mathbf{P}}_n^1 & 0 & \dots & 0 & 0 \\ 0 & \tilde{\mathbf{P}}_n^2 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \tilde{\mathbf{P}}_n^{M-1} & 0 \\ 0 & 0 & \dots & 0 & \tilde{\mathbf{P}}_n^M \end{bmatrix} \quad (61)$$

Podmacierze $\tilde{\mathbf{P}}_n^k$ ($k = 1, 2, \dots, M$) są macierzami kowariancji związanymi z parametrami adaptacyjnymi k -tego neuronu. Liczba parametrów macierzy \mathbf{P}_n wynosi $n \cdot M \times n \cdot M$ (n to rozmiar przestrzeni wejściowej, a M liczba neuronów warstwy ukrytej), natomiast macierz $\tilde{\mathbf{P}}_n$ ma $m^2 M$ istotnych parametrów. Tym samym dla danego problemu \mathcal{P} złożoność macierzy \mathbf{P}_n z $O(M^2)$ redukuje się do $O(M)$ dla

macierzy $\tilde{\mathbf{P}}_n$ (m jest stałe ze względu na \mathcal{P}). Powyższa redukcja prowadzi również do przedefiniowania równań (54–59) filtra EKF do postaci:

$$e_n = y_n - f(\mathbf{x}_n; \mathbf{p}_{n-1}) \quad i = 1, \dots, M \quad (62)$$

$$\mathbf{d}_n^i = \frac{\partial f(\mathbf{x}_n; \mathbf{p}_{n-1})}{\partial \mathbf{p}_{n-1}^i} \quad (63)$$

$$R_y = R_n + \mathbf{d}_n^{1T} \tilde{\mathbf{P}}_{n-1}^1 \mathbf{d}_n^1 + \dots + \mathbf{d}_n^{MT} \tilde{\mathbf{P}}_{n-1}^M \mathbf{d}_n^M \quad (64)$$

$$\mathbf{k}_n^i = \tilde{\mathbf{P}}_{n-1}^i \mathbf{d}_n^i / R_y \quad (65)$$

$$\mathbf{p}_n^i = \mathbf{p}_{n-1}^i + e_n \mathbf{k}_n^i \quad (66)$$

$$\tilde{\mathbf{P}}_n^i = [\mathbf{I} - \mathbf{k}_n^i \mathbf{d}_n^{iT}] \tilde{\mathbf{P}}_{n-1}^i + Q_0(n) \mathbf{I} \quad (67)$$

Kryterium wystarczalności modelu.

Kiedy aktualna architektura sieci nie jest już wystarczająca, aby akumulować nowe informacje? Odpowiedź na to pytanie nie jest trywialna. Powiększanie sieci o kolejne wagi czy neurony co pewien czas jest naiwnym podejściem. Kryteria, których wyznaczenie wymaga przejrzenia (przeanalizowania) zachowania modelu dla wszystkich wektorów treningowych, można realizować jedynie od czasu do czasu w trakcie procesu adaptacyjnego (raczej nie częściej niż co epokę). Algorytmy bazujące na wielkości popełnionego błędu dla danego wektora treningowego również nie są pozbawione wad, ponieważ podczas procesu uczenia nie możemy w pełni ufać modelowi, który podlega uczeniu — sieć nie jest w pełni wiarygodna. Kryterium wystarczalności powinno na jednej szali położyć wielkość błędu, a na drugiej stopień naszego zaufania do aktualnego stanu sieci:

$$\frac{\text{błąd}}{\text{niepewność modelu}} < \alpha(M) \quad (68)$$

$\alpha(M)$ jest progiem zależnym od wielkości struktury modelu (liczby stopni swobody).

Statystyczną miarą niepewności dla *catości* rozpatrywanego modelu jest jego wariancja, która składa się z wariancji modelu (sieci neuronowej) i szumu danych (np. niedoskonałości pomiarów). Przyjmijmy iż wariancja szumu danych jest równa σ_{ns}^2 , a wariancja samego modelu $Var(f(\mathbf{x}; \mathbf{p})) = \sigma_f^2(\mathbf{x})$.

Zakładając, iż błąd interpolacji pewnego odwzorowania ma rozkład normalny, można oszacować, czy błąd leży w pewnym przedziale ufności, wyznaczonym przez niepewność modelu i szumu danych z ustalonym stopniem zaufania. Hipotezę \mathcal{H}_0 , iż aktualny model jest wystarczający, można zapisać jako:

$$\mathcal{H}_0 : \frac{e^2}{Var[f(\mathbf{x}; \mathbf{p}) + \eta]} = \frac{e^2}{\sigma_f^2(\mathbf{x}) + \sigma_{ns}^2} < \chi_{M, \theta}^2 \quad (69)$$

gdzie $\chi_{M, \theta}^2$ jest rozkładem chi-kwadrat z progiem ufności $\theta\%$ i M stopniami swobody (M – liczba funkcji bazowych). Z kolei $e = y - f(\mathbf{x}; \mathbf{p})$ (por. (54)).

Gdy hipoteza \mathcal{H}_0 jest spełniona bieżąca struktura sieci jest wystarczająca przy ustalonym stopniu zaufania. W przeciwnym przypadku należy rozszerzyć pojemność modelu, czyli dodać nową funkcję bazową do warstwy ukrytej.

Używając filtra EKF do estymacji parametrów modelu mamy automatycznie wyznaczoną całkowitą wariancję modelu:

$$R_y = Var[f(\mathbf{x}; \mathbf{p}) + \sigma_{ns}^2], \quad (70)$$

R_y wyznaczone jest równaniem (58). Prowadzi to do ostatecznej definicji wystarczalności modelu:

$$\mathcal{H}_0 : \frac{e_n^2}{R_y} < \chi_{n,\theta}^2 \quad (71)$$

Jeśli hipoteza \mathcal{H}_0 jest spełniona, sieć IncNet kontynuuje proces adaptacji parametrów, używając do tego filtru EKF (lub jego szybkiej wersji). W przeciwnym przypadku należy dodać nową funkcję bazową $G_{M+1}(\cdot)$ (neuron) do warstwy ukrytej z pewnymi wartościami początkowymi. Dla funkcji bicentralnych mamy:

$$w_{M+1} = e_n = y_n - f(\mathbf{x}_n; \mathbf{p}_n) \quad (72)$$

$$\mathbf{t}_{M+1} = \mathbf{x}_n \quad (73)$$

$$\mathbf{b}_{M+1} = \mathbf{b}_0 \quad (74)$$

$$\mathbf{s}_{M+1} = \mathbf{s}_0 \quad (75)$$

$$\mathbf{P}_n = \begin{bmatrix} \mathbf{P}_n & 0 \\ 0 & \mathbf{P}_0 \mathbf{I} \end{bmatrix} \quad (76)$$

wektory stałych \mathbf{b}_0 i \mathbf{s}_0 definiują początkowe wartości rozmyć i skosów w wielowymiarowej przestrzeni.

Taki test można stosować w każdej iteracji algorytmu uczenia, tj. dla każdej prezentacji wektora treningowego. Jak się okazało w praktyce tak zdefiniowany test wystarczalności prowadzi do lepszego wykorzystania posiadanych już funkcji bazowych przez model, jak i do uzyskania lepszego poziomu generalizacji. Widać to porównując sieć IncNet z siecią RAN [47], czy RAN-EKF [42], co można zauważyć w pracy [35] jak i pracach [34, 36, 39].

Usuwanie neuronów. Podczas procesu uczenia często dochodzi do sytuacji, w której pewne wagi czy neurony przestają odgrywać istotną rolę. Model prowadzi nadal adaptację takich neuronów pomimo, iż nie są one przydatne, a nawet mogą być przyczyną przeuczenia się modelu.

Poniżej przedstawiono algorytm usuwania neuronów oparty o analizę wartości współczynników istotności (przydatności) dla neuronów sieci. Zakładając, iż wszystkie funkcje transferu neuronów warstwy ukrytej są podobne (np. zlokalizowane i wypukłe), współczynniki istotności poszczególnych neuronów warstwy ukrytej można zdefiniować przez stosunek wielkości wagi do wariancji tej wagi. Taki stosunek faworyzuje silne wagi (tj. wagi o istotnym wpływie), których tendencje zmian wartości są małe (tj. wagi *nauczone*). W ten sposób zdefiniowane współczynniki można zapisać matematycznie:

$$s_i = \frac{w_i^2}{\sigma_{w_i}} \quad (77)$$

gdzie σ_{w_i} oznacza wariancję i -tej wagi. Oczywiście jeśli miałyby dojść do usunięcia jakiegokolwiek neuronu, to należy wybrać ten, dla którego współczynnik istotności będzie najmniejszy:

$$L_1 = \min_i s_i = \min_i \frac{w_i^2}{\sigma_{w_i}} \quad (78)$$

Używając rozszerzonego filtra Kalmana jako estymatora parametrów modelu do wyznaczania wariancji σ_{w_i} , można użyć macierzy kowariancji \mathbf{P}_n . W tym celu najpierw przyjmijmy, iż parametry sieci w wektorze parametrów p_n są uszeregowane w poniższy sposób:

$$\mathbf{p}_n = [w_1, \dots, w_M, \dots]^T \quad (79)$$

czyli pierwsze są wagi pomiędzy warstwą drugą i wyjściową, a pozostałe parametry znajdują się w dalszej części wektora p_n . Wtedy macierz kowariancji \mathbf{P}_n wygląda tak:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_w & \mathbf{P}_{wv} \\ \mathbf{P}_{wv}^T & \mathbf{P}_v \end{bmatrix} \quad (80)$$

gdzie \mathbf{P}_w jest macierzą kowariancji pomiędzy wagami, macierz \mathbf{P}_{wv} jest macierzą kowariancji pomiędzy wagami i innymi parametrami, a macierz \mathbf{P}_v jest macierzą kowariancji tylko pomiędzy innymi parametrami (tj. bez wag). Korzystając z powyższego ułożenia parametrów w macierzy \mathbf{P} wzór (78) można sprowadzić do postaci:

$$L_1 = \min_i s_i = \min_i \frac{w_i^2}{[\mathbf{P}_w]_{ii}} \quad (81)$$

Oczywiście wartość L_1 można wyznaczać dla rozszerzonego filtra Kalmana, jak i dla jego szybkiej wersji, opisaną równaniami (62–67).

Pozostaje problem podjęcia decyzji, czy w ogóle należy usunąć jakiś neuron w danej k -tej iteracji algorytmu? Najczęściej nieistotne neurony występują w modelach niestabilnych, a wartościowe neurony mamy w dobrze nauczonych, stabilnych sieciach. Ogólnie kryterium można zapisać jako

$$\frac{L_1}{R_y} < \chi_{1,\vartheta}^2 \quad (82)$$

gdzie $\chi_{1,\vartheta}^2$ jest rozkładem chi-kwadrat z poziomem ufności $\vartheta\%$ z jednym stopniem swobody, a R_y jest całkowitą wariancją modelu, która w przypadku użycia filtra Kalmana jest wyznaczana wzorem (58). Jeśli powyższe kryterium jest spełnione oznacza to, że należy dokonać usunięcia neuronu, dla którego uzyskano najmniejszą wartość spośród współczynników istotności.

Tak zdefiniowane statystyczne kryterium usuwania neuronów warstwy ukrytej dla sieci typu RBF może być stosowane w praktyce z iteracji na iterację, szczególnie, gdy prowadzimy estymacje parametrów sieci za pomocą rozszerzonego filtra EKF. Podczas uczenia sieci, kontroli wystarczalności i kontroli przydatności poszczególnych neuronów, model na bieżąco stara się dopasować złożoność struktury sieci do złożoności napływających sekwencyjnie danych uczących, co dobrze wpływa na końcowy poziom generalizacji.

Współpracujące statystyczne kryteria wystarczalności sieci i przydatności neuronów są używane w sieci IncNet, a badania, wykonane dla różnych danych, wykazują dużą skuteczność ich działania [36, 37, 31, 34, 33, 32]. Rezultaty te zostały szeroko opisane w [35].

Łączenie neuronów. Nierzadko mamy do czynienia z sytuacją, w której regiony decyzyjne, powstałe przez kooperację wielu neuronów, mogłyby zostać zastąpione przez prostszą sieć, nie zmniejszając właściwości klasyfikacji, czy aproksymacji, a nawet dając możliwość polepszenia jakości generalizacji, dzięki zbliżeniu złożoności modelu adaptacyjnego do złożoności danych uczących (problemu). Dlatego też możliwość łączenia dwóch (lub większej liczby) neuronów może wpłynąć pozytywnie, nie zmieniając przy tym istotnie powierzchni aproksymowanej funkcji, czy powierzchni regionów decyzyjnych dla klasyfikacji.

Połączenia dwóch neuronów i, j oraz zastąpienia ich przez nowy neuron new , można dokonać przy założeniu, iż funkcje transferu neuronów są zlokalizowane, ciągłe i jest spełnione poniższe kryterium:

$$\frac{\int_{d \subseteq \mathcal{D}^n} |\bar{\phi}_i(\mathbf{x}) + \bar{\phi}_j(\mathbf{x}) - \bar{\phi}_{new}(\mathbf{x})| d\mathbf{x}}{\int_{d \subseteq \mathcal{D}^n} |\bar{\phi}_i(\mathbf{x}) + \bar{\phi}_j(\mathbf{x})| d\mathbf{x}} < \alpha \quad (83)$$

gdzie d jest podprzestrzenią aktywności funkcji transferu $\phi_i(\mathbf{x})$ i $\phi_j(\mathbf{x})$, a $\bar{\phi}_i(\mathbf{x}) = w_i \phi_i(\mathbf{x})$ i $\bar{\phi}_j(\mathbf{x}) = w_j \phi_j(\mathbf{x})$ są funkcjami przeskalowanymi przez wagi wyjściowe. Przeskalowana funkcja transferu nowego neuronu $\bar{\phi}_{new}(\mathbf{x})$ ($\bar{\phi}_{new}(\mathbf{x}) = w_{new} \phi_{new}(\mathbf{x})$) dla \mathbf{x} spoza podprzestrzeni d musi być w przybliżeniu równa zero. Współczynnik α określa błąd, jaki uznaje się za dopuszczalny. Współczynnik α uzależnić liniowo od sumy pomiarów R_n lub od całkowitej wariancji modelu R_y (patrz równanie 58).

Kryterium opisane równaniem (83) jest trudno zastosować w ogólnym przypadku, lecz gdy funkcje transferu warstwy ukrytej są separowalne wymiarowo (na przykład dla funkcji gaussowskich lub bicentralnych), można dokonać uproszczenia tego kryterium do postaci:

$$\frac{\int_{d_1 \subseteq \mathcal{D}_1} \dots \int_{d_N \subseteq \mathcal{D}_N} [\bar{\phi}_i(\mathbf{x}) + \bar{\phi}_j(\mathbf{x}) - \bar{\phi}_{new}(\mathbf{x})]^2 dx_1 \dots dx_N}{\int_{d_1 \subseteq \mathcal{D}_1} \dots \int_{d_N \subseteq \mathcal{D}_N} [\bar{\phi}_i(\mathbf{x}) + \bar{\phi}_j(\mathbf{x})]^2 dx_1 \dots dx_N} < \alpha \quad (84)$$

Powyższe kryterium może zostać użyte w formie analitycznej lub numerycznej. W innych przypadkach kryterium łączenia neuronów może zostać uproszczone do wyznaczania ważonego błędu średniokwadratowego w oparciu o zbiór punktów o gaussowskim rozkładzie, rozmieszczonych na obszarze aktywności neuronów i, j :

$$\frac{\sum_{d \in \mathbf{d}} [\bar{\phi}_i(\mathbf{x}) + \bar{\phi}_j(\mathbf{x}) - \bar{\phi}_{new}(\mathbf{x})]^2}{\sum_{d \in \mathbf{d}} [\bar{\phi}_i(\mathbf{x}) + \bar{\phi}_j(\mathbf{x})]^2} < \alpha \quad (85)$$

W przypadku użycia funkcji bicentralnych jako funkcji transferu, poszczególne parametry funkcji bicentralnej mogą być wyznaczone jak poniżej:

$$\mathbf{w}_{new} = \frac{\bar{\phi}(\mathbf{t}_i, \mathbf{t}_i) \cdot \bar{\mathbf{P}}_i + \bar{\phi}(\mathbf{t}_j, \mathbf{t}_j) \cdot \bar{\mathbf{P}}_j}{\bar{\phi}(\mathbf{t}_{new}, \mathbf{t}_{new})} \quad (86)$$

$$\mathbf{t}_{new, k} = \frac{1}{M} \int_{d \in \mathbf{D}} x_k [\bar{\phi}(\mathbf{x}, \mathbf{t}_i) + \bar{\phi}(\mathbf{x}, \mathbf{t}_j)] d\mathbf{x} \quad (87)$$

$$\mathbf{s}_{new} = \mathbf{s}_i \cdot \bar{\mathbf{P}}_i + \mathbf{s}_j \cdot \bar{\mathbf{P}}_j \quad (88)$$

$$\mathbf{b}_{new} = \begin{cases} \mathbf{b}_i & \text{neuron } j \text{ wewnątrz neuronu } i \\ \mathbf{b}_j & \text{neuron } i \text{ wewnątrz } j \\ \frac{\mathbf{b}_i + \mathbf{b}_j + |\mathbf{t}_i - \mathbf{t}_j|}{2} & \text{w p. p.} \end{cases} \quad (89)$$

gdzie M jest zdefiniowane przez

$$\mathbf{M} = \int_{d \in \mathbf{D}} [\bar{\phi}(\mathbf{x}, \mathbf{t}_i) + \bar{\phi}(\mathbf{x}, \mathbf{t}_j)] d\mathbf{x} = \mathbf{P}_i + \mathbf{P}_j \quad (90)$$

natomiast $\bar{\mathbf{P}}_i$ i $\bar{\mathbf{P}}_j$ poprzez:

$$\bar{\mathbf{P}}_i = \frac{\mathbf{P}_i}{(\mathbf{P}_i + \mathbf{P}_j)} \quad \bar{\mathbf{P}}_j = \frac{\mathbf{P}_j}{(\mathbf{P}_i + \mathbf{P}_j)} \quad (91)$$

gdzie \mathbf{P}_i i \mathbf{P}_j są zdefiniowane jako

$$\mathbf{P}_i = \int_{d \in \mathbf{D}} \bar{\phi}(\mathbf{x}, \mathbf{t}_i) d\mathbf{x} \quad \mathbf{P}_j = \int_{d \in \mathbf{D}} \bar{\phi}(\mathbf{x}, \mathbf{t}_j) d\mathbf{x} \quad (92)$$

Pozostaje pytanie, kiedy próbować, czy kryterium będzie spełnione, i dla jakich par neuronów sprawdzać, czy kryterium jest spełnione. Jednym ze sposobów jest sprawdzanie kryterium co epokę dla każdego neuronu i ($i = 1, \dots, M$) i neuronu j , wybranego w następujący sposób:

$$j = \arg \max_k \phi(\mathbf{t}_k, \mathbf{t}_i) \quad (93)$$

Innym i kosztowniejszym sposobem jest próba łączenia jednej pary neuronów podczas każdej (p -tej) iteracji algorytmu adaptacji. W tym przypadku wybiera się pierwszy neuron i :

$$i = \arg \max_k \phi(\mathbf{x}_p, \mathbf{t}_k) \quad (94)$$

gdzie \mathbf{x}_p jest wektorem wejściowym prezentowanym w p -tej iteracji algorytmu. Następnie wyznacza się drugi neuron j :

$$j = \arg \max_{k \neq i} \phi(\mathbf{x}_p, \mathbf{t}_k) \quad (95)$$

po czym bada się, czy jest spełnione kryterium łączenia neuronów dla neuronu i, j . Gdy, dla pewnej pary neuronów kryterium łączenia będzie spełnione (niezależnie od tego czy sprawdzanie następuje co iterację, czy co epokę), następuje zastąpienie owej pary neuronów nowym neuronem o parametrach opisanych wzorami (86–89).

Powyżej zaproponowany sposób kontroli złożoności umożliwia zmniejszenie struktury sieci neuronowej poprzez unifikację jej fragmentów. Dzięki tej metodzie nierzadko da się zastąpić dwa neurony, które odgrywają istotną rolę, jednym neuronem nie tracąc jakości działania modelu. Tak zdefiniowaną metodę łączenia neuronów można stosować praktycznie do każdej sieci typu RBF, jak do innych modeli opartych o ciągłe i zlokalizowane funkcje transferu.

Wykorzystanie sieci IncNet w klasyfikacji Opisana powyżej sieć IncNet ma tylko jedno wyjście. Adaptując wagi pomiędzy warstwą ukrytą i wyjściową można sformułować sposób uczenia sieci z wieloma wyjściami (zrobił to Kadirkamanathan w pracy [41]). Jednakże rezygnacja z adaptacji położenia funkcji bazowych, rozmycie czy skosów i ewentualnie innych parametrów dla funkcji bicentralnych zmniejsza jego potencjalne możliwości modelu. Z kolei liniowy układ wyjścia (klasa 1, 2, ..., K)

jest niedopuszczalnym rozwiązaniem, gdyż nie odzwierciedla w żaden sposób rzeczywistych zależności pomiędzy klasami.

Z powyższych powodów najciekawszym rozwiązaniem wydaje się pomysł zbudowania klastra niezależnych sieci IncNet. Zadaniem każdej z podsieci jest wyspecjalizowanie się w rozpoznawaniu jednej (k -tej) klasy. W tym celu ze zbioru par uczących:

$$\mathcal{S} = \{\langle \mathbf{x}_1, y_1 \rangle, \langle \mathbf{x}_2, y_2 \rangle, \dots, \langle \mathbf{x}_N, y_N \rangle\} \quad (96)$$

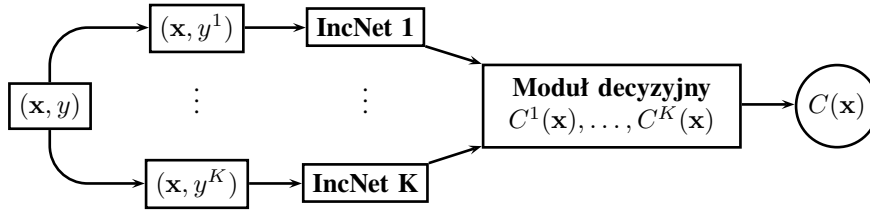
produkujemy K zbiorów, po jednym dla każdej klasy:

$$\mathcal{S}^k = \{\langle \mathbf{x}_1, y_1^k \rangle, \langle \mathbf{x}_2, y_2^k \rangle, \dots, \langle \mathbf{x}_N, y_N^k \rangle\} \quad k = 1, 2, \dots, K \quad (97)$$

gdzie y_i^k przyjmuje wartości 1 lub 0:

$$y_i^k = \begin{cases} 1 & y_i = k \\ 0 & y_i \neq k \end{cases} \quad i = 1, 2, \dots, N \quad (98)$$

Tak zdefiniowane zbiory \mathcal{S}^k są zbiorami uczącymi dla poszczególnych sieci IncNet, które razem tworzą klastrowanie sieci o schemacie przedstawionym na rysunku 4.



Rysunek 4: Klastrowanie sieci IncNet z zastosowaniem do problemów klasyfikacyjnych.

Dla danej obserwacji \mathbf{x} każda z sieci produkuje wartość wyjściową $C^i(\mathbf{x})$ (dla $i = 1, 2, \dots, K$). Wartość $C^i(\mathbf{x})$ zbliżona do zera oznacza, iż obserwacja \mathbf{x} nie odpowiada klasie i . Natomiast wartość $C^i(\mathbf{x})$ zbliżona do jedynki oznacza, iż obserwacja \mathbf{x} odpowiada klasie i .

Moduł decyzyjny, widoczny na rysunku 4, podejmuje ostateczną decyzję i przypisuje pewien wektor \mathbf{x} do jednej klasy $C(\mathbf{x})$. Podejmowanie takiej decyzji może przebiegać zgodnie z poniższą definicją:

$$C(\mathbf{x}) = \arg \max_i C^i(\mathbf{x}) \quad (99)$$

Tym samym moduł decyzyjny wybiera sieć, której aktywacja była największa, tj. najbardziej odpowiadająca danej obserwacji.

Jednakże przydatność takiego klastra podsieci IncNet wcale nie musi sprowadzać się do obserwacji jedynie wartości $C(\mathbf{x})$. Bardzo przydatne jest wprowadzenie następującej renormalizacji:

$$p(C^i|\mathbf{x}) = \frac{\sigma(C^i(\mathbf{x}) - \frac{1}{2})}{\sum_{j=1}^K \sigma(C^j(\mathbf{x}) - \frac{1}{2})} \quad (100)$$

gdzie $\sigma(x) = 1/(1 + \exp(-\gamma x))$, a γ jest stałą (np. około 10). Prowadzi to do aproksymacji prawdopodobieństwa przynależności wektora \mathbf{x} do klasy i .

5 Przykładowe zastosowanie sieci IncNet

Poniżej będzie można prześledzić sprawność powyżej przedstawionych kryteriów kontroli złożoności sieci neuronowych. Analizie zostaną poddane dane psychometryczne. Liczba wektorów w wybranej bazie wynosiła 1027 przypadków. Każdy przypadek jest opisywany przez 14 różnych cech opisujących pewne skale kontrolne i kliniczne. Skale kontrolne i kliniczne są wyznaczane na podstawie testu MMPI [3, 2]. Celem jest klasyfikacja danego przypadku do jednej z 27 klas nozologicznych (norma, nerwica, organika, schizofrenia, psychopatia, zespół urojeniowy, etc.).

Sieć IncNet wykorzystywana do problemów klasyfikacji, składa się z klastra podsieci, a zadaniem każdej z podsieci jest estymacja prawdopodobieństwa przynależności do każdej z klas niezależnie. Poniższa tabela prezentuje, jak rozkłada się liczba neuronów sieci IncNet w poszczególnych podsieciach. Tabela zawiera informacje uzyskane na podstawie uczenia na zbiorze 27 klasowym. Proces uczenia sieci IncNet trwał 5 epok. Poprawność klasyfikacji wynosiła 99.22%.

Liczby neuronów w poszczególnych podsieciach																										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
5	1	4	9	9	6	4	3	8	8	3	11	4	1	8	4	5	8	12	8	1	2	2	1	1	1	1
Całkowita liczba neuronów: 130																										

Jak widać dysproporcje pomiędzy złożonością poszczególnych podsieci są znaczne. Podobne zachowania obserwuje się w większości realnych baz danych. Zmienność liczby neuronów w procesie uczenia, jak i zmianę wartości błędu treningowego i testowego, dla kilku wybranych podsieci, można przeanalizować na kolejnych rysunkach. Widzimy na nich zmiany, które zostały zebrane w 25 punktach kontrolnych (czyli co około 200 iteracji, każda sieć była uczona 5 epok).

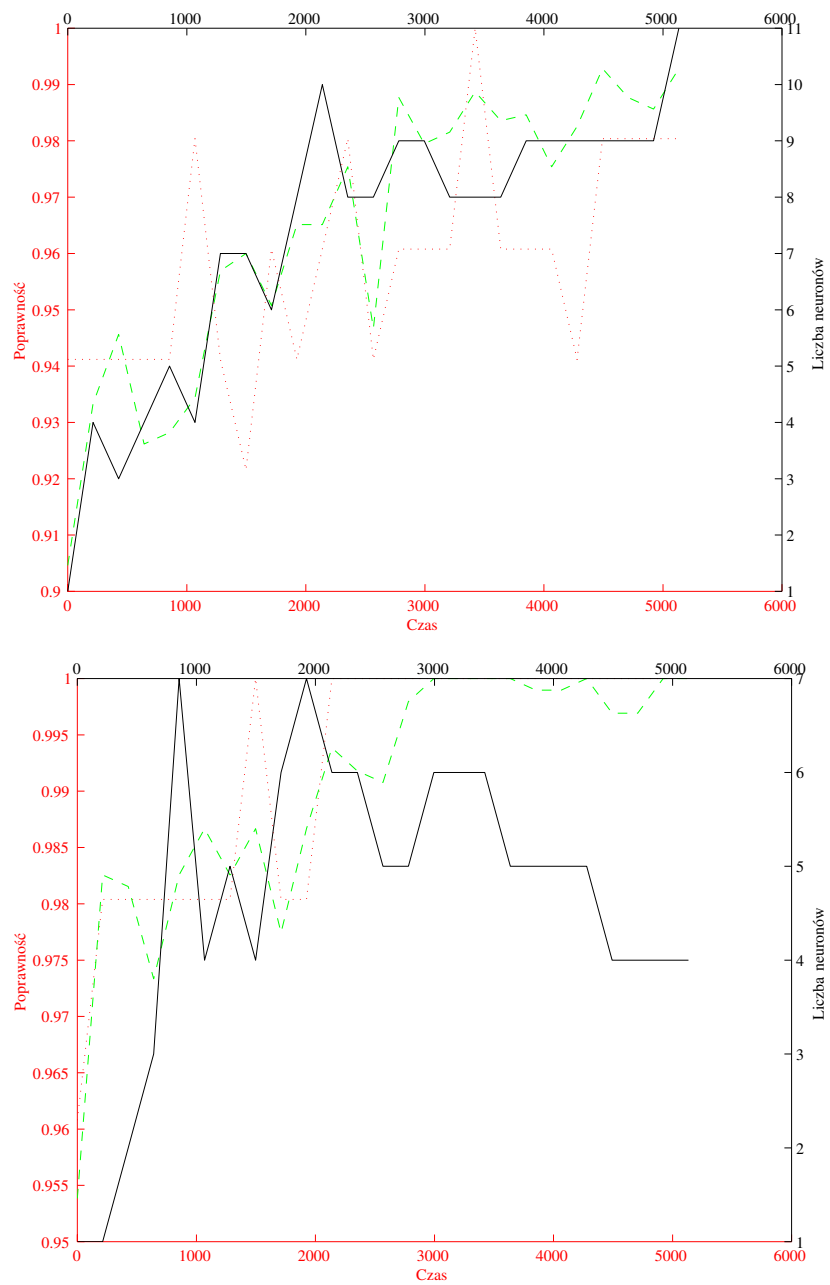
Rysunek 5 przedstawia przykładowy proces uczenia dla 5-tej i 16-tej klasy. Należy zwrócić uwagę, że jednostką czasu jest tu jedna iteracja, czyli prezentacja jednego wektora treningowego. Ciągła krzywa obrazuje liczbę neuronów, linie kropkowana i przerywana pokazują poprawność klasyfikacji dla zbioru treningowego i testowego. Dokładniejszą analizę opisanych danych, jak i inne przykłady zastosowań sieci ontogenicznych można znaleźć w [35].

6 Podsumowanie

Przedstawiono tu przegląd metod, które kontrolują złożoność sieci starając się wspomagać proces adaptacji tak, aby był on jak najskuteczniejszy. Większa część metod umożliwia powiększanie struktury sieci albo jej pomniejszanie, ale są i takie metody, które są zdolne do powiększania i zmniejszania swojej struktury. Część metod jest zdolna do prowadzenia bieżącej kontroli złożoności, inne z kolei robią to co epokę, po uzyskaniu zbieżności lub po procesie adaptacji.

Opisany szczegółowo model sieci IncNet stosuje efektywne statystyczne kryteria kontroli złożoności.

Przykład opisany w poprzednim rozdziale, zastosowania sieci IncNet, ilustruje, że złożoność poszczególnych części problemu może być bardzo różna i trudna do określenia przed procesem uczenia. Tym samym trudno jest ocenić przed procesem uczenia złożoność całego problemu. Z drugiej strony, zły dobór złożoności architektury sieci i tym samym jej nieadekwatność w stosunku do złożoności problemu



Rysunek 5: Wykres ilustruje zmieniającą się w czasie poprawność klasyfikacji dla zbioru treningowego (kolor zielony) i zbioru testowego (kolor czerwony), jak i liczbę neuronów (kolor czarny). Dane dla zbioru 27-klasowego, klasy 5-tej i 16-tej. [Jednostką czasu jest prezentacja pojedynczego wektora.]

może doprowadzić do zupełnie nieprzydatnych rezultatów — brak zadowalającej generalizacji. Wszystko to prowadzi do konkluzji, że kontrola złożoności powinna być wbudowana w mechanizm uczenia i powinna działać możliwie sprawnie.

Bibliografia

- [1] R. Adamczak, W. Duch, N. Jankowski. New developments in the feature space mapping model. *Third Conference on Neural Networks and Their Applications*, s. 65–70, Kule, Poland, 1997.
- [2] Y. S. Ben-Porath, N. Sherwood. *The MMPI-2 content component scales: Development, psychometric characteristics, and clinical applications*. University of Minnesota Press, Minneapolis, 1993.
- [3] J. N. Butcher, W. G. Dahlstrom, J. R. Graham, A. Tellegen, B. Kaemmer. *Minnesota Multiphasic Personality Inventory-2 (MMPI-2): Manual for administration and scoring*. University of Minnesota Press, Minneapolis, 1989.
- [4] J. V. Candy. *Signal processing: The model based approach*. McGraw-Hill, New York, 1986.
- [5] M. Cottrell, B. Girard, Y. Girard, M. Mangeas, C. Muller. Neural modeling for time series: a statistical stepwise method for weight elimination. *IEEE Transaction on Neural Networks*, 6(6):1355–1364, 1995.
- [6] Y. Le Cun, B. Boser, J. Denker, D. Henderson, W. Hubbard, L. Jackel. Handwritten digit recognition with a back-propagation network. D. S. Touretzky, redaktor, *Advances in Neural Information Processing Systems 2*, San Mateo, CA, 1990. Morgan Kaufman.
- [7] Y. Le Cun, J. Denker, S. Solla. Optimal brain damage. D. S. Touretzky, redaktor, *Advances in Neural Information Processing Systems 2*, San Mateo, CA, 1990. Morgan Kaufman.
- [8] W. Duch, R. Adamczak, K. Grąbczewski. Extraction of logical rules from backpropagation networks. *Neural Processing Letters*, 7:1–9, 1998.
- [9] W. Duch, R. Adamczak, N. Jankowski. Initialization of adaptive parameters in density networks. *Third Conference on Neural Networks and Their Applications*, s. 99–104, Kule, Poland, 1997.
- [10] W. Duch, R. Adamczak, N. Jankowski. New developments in the feature space mapping model. Raport techniczny CIL-KMK-2/97, Computational Intelligence Lab, DCM NCU, Toruń, Poland, 1997. (long version).
- [11] W. Duch, G. H. F. Dierksen. Feature space mapping as a universal adaptive system. *Computer Physics Communications*, 87:341–371, 1994.
- [12] W. Duch, N. Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, (2):163–212, 1999. (PDF).
- [13] W. Duch, N. Jankowski, A. Naud, R. Adamczak. Feature space mapping: a neurofuzzy network for system identification. *Proceedings of the European Symposium on Artificial Neural Networks*, s. 221–224, Helsinki, 1995.

- [14] S. E. Fahlman. The recurrent cascade-correlation architecture. Raport techniczny, Carnegie Mellon University, School of Computer Science, 1991.
- [15] S. E. Fahlman, C. Lebiere. The cascade-correlation learning architecture. D. S. Touretzky, redaktor, *Advances in Neural Information Processing Systems 2*, s. 524–532. Morgan Kaufmann, 1990.
- [16] S. E. Fahlman, C. Lebiere. The cascade-correlation learning architecture. Raport techniczny CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1990.
- [17] E. Fiesler. Comparative bibliography of ontogenic neural networks. *Proceedings of the International Conference on Artificial Neural Networks*, 1994.
- [18] W. Finnoff, F. Hergert, H. G. Zimmermann. Improving model detection by nonconvergent methods. *Neural Networks*, 6(6):771–783, 1993.
- [19] W. Finnoff, H. G. Zimmermann. Detecting structure in small datasets by network fitting under complexity constrains. *Proceedings of the second annual workshop on computational learning theory and natural learning systems*, Berkeley, CA, 1991.
- [20] M. Freat. The upstart algorithm: a method for constructing and training feed-forward neural networks. *Neural Computation*, 2(2):198–209, 1990.
- [21] B. Fritzke. Fast learning with incremental RBF networks. *Neural Processing Letters*, 1(1):2–5, 1994.
- [22] B. Fritzke. Supervised learning with growing cell structures. J. D. Cowan, G. Tesauro, J. Alspector, redaktorzy, *Advances in Neural Information Processing Systems 6*, San Mateo, CA, 1994. Morgan Kaufman.
- [23] B. Fritzke. A growing neural gas network lerns topologies. G. Tesauro, D. S. Touretzky, T. K. Leen, redaktorzy, *Advances in Neural Information Processing Systems 7*, Cambridge, MA, 1995. MIT Press.
- [24] B. Fritzke. A self-organizing network that can follow non-stationary distributions. W. Gerstner, A. Germond, M. Hasler, J. Nicoud, redaktorzy, *7th International Conference on Artificial Neural Networks*, s. 613–618, Lausanne, Switzerland, 1997. Springer-Verlag.
- [25] S. I. Gallant. Three constructive algorithms for network learning. *Proceedings of the eighth annual conference of the cognitive science society*, s. 652–660, Hillsdale, NJ, 1986. Lawrence Erlbaum.
- [26] G. H. Golub, M. Heath, G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–213, 1979.
- [27] B. Hassibi, D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993.
- [28] B. Hassibi, D. G. Stork, G. J. Wolff. Optimal brain surgeon and general network pruning. Raport techniczny CRC-TR-9235, RICOH California Research Center, Menlo Park, CA, 1992.

- [29] S. Haykin. *Adaptive filter theory*. Printice-hall international, 1996.
- [30] G. E. Hinton. Learning translation invariant recognition in massively parallel networks. J. W. de Bakker, A. J. Nijman, P. C. Treleaven, redaktorzy, *Proceedings PARLE Conference on Parallel Architectures and Languages Europe*, s. 1–13, Berlin, 1987. Springer-Verlag.
- [31] N. Jankowski. Controlling the structure of neural networks that grow and shrink. *Second International Conference on Cognitive and Neural Systems*, Boston, USA, 1998.
- [32] N. Jankowski. Approximation and classification in medicine with IncNet neural networks. *Machine Learning and Applications. Workshop on Machine Learning in Medical Applications*, s. 53–58, Greece, 1999. (PDF).
- [33] N. Jankowski. Approximation with RBF-type neural networks using flexible local and semi-local transfer functions. *4th Conference on Neural Networks and Their Applications*, s. 77–82, Zakopane, Poland, 1999. (PDF).
- [34] N. Jankowski. Flexible transfer functions with ontogenic neural. Raport techniczny, Computational Intelligence Lab, DCM NCU, Toruń, Poland, 1999. (PDF).
- [35] N. Jankowski. *Ontogeniczne sieci neuronowe w zastosowaniu do klasyfikacji danych medycznych*. Praca doktorska, Katedra Metod Komputerowych, Uniwersytet Mikołaja Kopernika, Toruń, Polska, 1999. (PDF).
- [36] N. Jankowski, V. Kadirkamanathan. Statistical control of growing and pruning in RBF-like neural networks. *Third Conference on Neural Networks and Their Applications*, s. 663–670, Kule, Poland, 1997.
- [37] N. Jankowski, V. Kadirkamanathan. Statistical control of RBF-like networks for classification. *7th International Conference on Artificial Neural Networks*, s. 385–390, Lausanne, Switzerland, 1997. Springer-Verlag.
- [38] V. Kadirkamanathan. *Sequential learning in artificial neural networks*. Praca doktorska, Cambridge University Engineering Department, 1991.
- [39] V. Kadirkamanathan. A statistical inference based growth criterion for the RBF networks. *Proceedings of the IEEE. Workshop on Neural Networks for Signal Processing*, 1994.
- [40] V. Kadirkamanathan, M. Niranjan. Nonlinear adaptive filtering in nonstationary environments. *Proceedings of the international conference on acoustic, speech and signal processing*, Toronto, 1991.
- [41] V. Kadirkamanathan, M. Niranjan. Application of an architecturally dynamic network for speech pattern classification. *Proceedings of the Institute of Acoustics*, 14:343–350, 1992.
- [42] V. Kadirkamanathan, M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5(6):954–975, 1993.
- [43] J. Korbicz, A. Obuchowicz, D. Uciński. *Sztuczne sieci neuronowe, podstawy i zastosowania*. Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1994.

- [44] M. Mezard, J. P. Nadal. Learning in feedforward layered networks: the tiling algorithm. *Journal of physics*, 22:2191–2203, 1989.
- [45] S. J. Nowlan, G. E. Hinton. Simplifying neural networks by soft weight sharing. *Neural Computation*, 4(4):473–493, 1992.
- [46] M. Orr. Introduction to radial basis function networks. Raport techniczny, Centre for Cognitive Science, University of Edinburgh, 1996.
- [47] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225, 1991.
- [48] L. Rutkowski. *Filtry adaptacyjne i adaptacyjne przetwarzanie sygnałów*. Wydawnictwa Naukowo–Techniczne, Warszawa, 1994.
- [49] A. S. Weigend, D. E. Rumelhart, B. A. Huberman. Back–propagation, weight elimination and time series prediction. *Proceedings of the 1990 Connectionist Models Summer School*, s. 65–80. Morgan Kaufmann, 1990.
- [50] A. S. Weigend, D. E. Rumelhart, B. A. Huberman. Generalization by weight elimination with application to forecasting. R. P. Lipmann, J. E. Moody, D. S. Touretzky, redaktorzy, *Advances in Neural Information Processing Systems 3*, s. 875–882, San Mateo, CA, 1991. Morgan Kaufmann.
- [51] A. Weigend, H. Zimmermann, Ralph Neuneier. Clearing. P. Refenes, Y. Abu-Mostafa, J. Moody, A. Weigend, redaktorzy, *Proceedings of NNCM, Neural Networks in Financial Engineering*, Singapore, 1995. World Scientific.