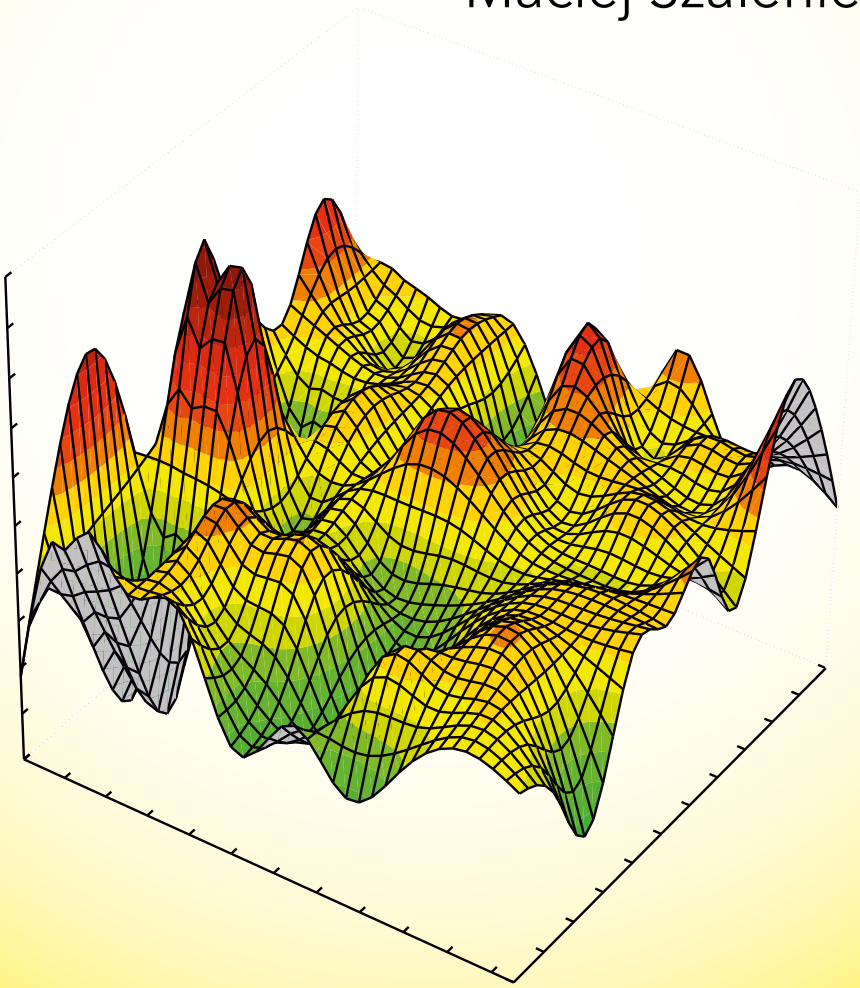
LEKSYKON SIECI NEURONOWYCH

Ryszard Tadeusiewicz
Maciej Szaleniec



Wydanie I
Wrocław 2015

Publikacja finansowana ze środków Ministerstwa Nauki i Szkolnictwa Wyższego w ramach programu Działalność Upowszechniająca Naukę.

Korekta:
Bartosz Ryż

Skład i łamanie e-publikacji:
Studio grafiki i DTP Grafpa

Projekt okładki:
autorzy

ISBN 978-83-63270-10-0

Wydawnictwo Fundacji „Projekt Nauka"
ul. Gwarecka 8/12
54-143 Wrocław
biuro@projekt-nauka.com
www.projekt-nauka.com

Ryszard Tadeusiewicz
Maciej Szaleniec

Leksykon sieci neuronowych

Wrocław 2015

Spis treści

O

P

Q

R

S

T

U

W

Z

Wprowadzenie

Sieci neuronowe, kiedyś awangardowe i fascynujące jako narzędzia informatyczne będące jednocześnie – oczywiście tylko w pewnym zakresie – modelami ludzkiego mózgu, są już zaledwie popularnymi technikami obliczeniowymi, wykorzystywanymi chętnie i często, ale nie budzącymi już dziś większych emocji. Z takim ściśle pragmatycznym podejściem do sieci neuronowych, traktowanych wyłącznie jako narzędzia do rozwiązywania praktycznych problemów, wiąże się też zmieniony model zapotrzebowania na wiedzę, która jest z nimi związana. Kiedyś czytelnicy byli na tyle zainteresowani samymi sieciami neuronowymi jako takimi, że bardzo poszukiwali książek na ten temat i skwapliwie z nich korzystali. Odwołajmy się do przykładu książki [1], która jako pierwsza w Polsce poświęcona tej tematyce miała tak wielkie powodzenie, że pierwszy jej nakład rozszedł się w ciągu dwóch tygodni i trzeba było jeszcze w tym samym roku przygotować i wydać wydanie drugie – oczywiście poprawione i uzupełnione. Książka ta była dość pilnie czytana, o czym świadczy ponad 500 udokumentowanych cytowań w Google Scholar i ponad 20 prac doktorskich, do opiniowania których zapraszano autora, ponieważ były one oparte na merytorycznej bazie tej właśnie książki. Warto jeszcze raz podkreślić: w tamtym pionierskim okresie przeczytanie nawet tak obszernej i miejscami bardzo szczegółowej książki nie wydawało się Czytelnikom wysiłkiem nadmiernym ani źle ulokowanym.

W kolejnych latach (1994 i 1995) pojawiły się liczne książki zagraniczne na temat sieci neuronowych, ale w Polsce ich przyrost nie był szczególnie duży: w 1994 roku książka [2], w 1995 książka [3]. Opiniując w tym czasie liczne artykuły oraz inne prace naukowe (na przykład doktoraty i habilitacje) dotykające problematyki sieci neuronowych, jeden z autorów tego *Leksykonu* miał okazję przekonać się, że wspomniane książki były bardzo dokładnie czytane, a uważni czytelnicy wychwytywali i wykorzystywali nawet bardzo drobne niuanse.

Potem jednak ruszyła lawina: w 1996 roku ukazały się już cztery książki poświęcone problematyce sieci neuronowych [3], [4], [5] i [6], a w kolejnych latach było ich już tak wiele, że trudno by je tu było wszystkie wymienić, nie mówiąc o jakimś omawianiu czy charakteryzowaniu ich treści.

Jednak, jak wspomniano, ta fascynacja sieciami neuronowymi jako takimi już minęła. Obecnie prawie nikt już nie zastanawia się nad tym, jakie cechy naturalnej (biologicznej) komórki nerwowej posiada sztuczny neuron będący składnikiem sieci rozwiązującej jakiś praktyczny problem. **Przedmiotem zainteresowania jest bowiem rozwiązywany problem, a sieć stała się tylko narzędziem służącym do uzyskania rozwiązania**. Takiemu podejściu sprzyja także łatwa dostępność różnych programów, oferujących sieci neuronowe jako łatwe w użyciu narzędzie informatyczne, typu „tu się wkłada dane, a tu się otrzymuje wyniki". Nieważne, co jest w środku, byle dane były dobrze wykorzystane a wynik był przydatny! Przykładem często używanych programów tego typu jest *Neural Networks Toolbox for Matlab* albo *Statistica Neural Networks*. Mniej znanym, ale też znakomitym profesjonalnym programem jest pakiet *Mathematica Neural Networks*, który jest dziełem genialnego matematyka i fizyka Stephena Wolframa.

Takiemu podejściu użytkowników towarzyszył także zmieniający się profil następnych wydawanych książek – zamiast opisu **sieci neuronowych** jako takich pojawiały się w nich głównie informacje na temat **zastosowań** tych sieci. A ponieważ tych zastosowań (udanych!) było bardzo dużo – pojawiały się też coraz obszerniejsze opracowania książkowe, gromadzące właśnie takie artykuły i doniesienia. Większość tych artykułów miała podobny schemat: „chcieliśmy rozwiązać problem X, zastosowaliśmy sieć neuronową oraz rekomendowaną przez innych autorów metodę Y, porównaliśmy wyniki i sieć neuronowa okazała się lepsza od metody Y". Jak się wydaje najbardziej okazałymi działami tego typu w Polsce były monografie [8] i [9], wydawane pod egidą Polskiej Akademii Nauk. Pierwsza z nich zawierała 26 rozdziałów i mieściła się na 833 stronicach, druga zawierała 27 rozdziałów (oraz obszerny dodatek) i mieściła się na 745 stronicach. Oczywiście tak obszernych książek nikt już nie czytał „od deski do deski", tylko każdy czytelnik takiej książki wydobywał z niej to, co go w danym momencie interesowało w kontekście jego problemu, do rozwiązania którego zamierzał użyć sieci neuronowych.

Jednak przy takim ściśle pragmatycznym i utylitarnym podejściu do sieci neuronowych pojawia się niekiedy trudność, polegająca na tym, że osoba czytająca o interesującym **zastosowaniu** tych sieci często napotyka w tekście studiowanego artykułu lub rozdziału monografii specja-

listyczny termin, związany właśnie ze specyficzną wiedzą na temat sieci neuronowych jako takich. Autor czytanego artykułu tego terminu nie objaśnia, gdyż dla niego (i dla innych osób, które dobrze poznały problematykę sieci neuronowych) jest to pojęcie elementarne, oczywiste, nie wymagające objaśnień. Inna jest jednak sytuacja czytelnika, który chce sieci użyć, ale brakuje mu owej specjalistycznej wiedzy. Oczywiście można w tym momencie powiedzieć, że skoro nie wie, to powinien się douczyć, sięgając do książek kompleksowo przedstawiających całość problematyki sieci neuronowych jako takich. Jednak taki postulat jest nierealistyczny. Ktoś kto chce szybko i skutecznie rozwiązać jakiś swój problem, prędzej porzuci narzędzie, jakim są sieci neuronowe, niż zdecyduje się na dokładne studiowania ich teorii. „Nie kupuje się browaru, gdy chce się wypić szklankę piwa" – to popularne powiedzenie dość wiernie oddaje istotę dylematu, przed którym stoi opisywany tutaj badacz lub praktyk.

Właśnie dla takich osób, chcących **używać** sieci neuronowych bez zagłębiania się w szczegóły wiedzy na ich temat – przeznaczona jest ta książka. *Leksykon* zbudowany został w taki sposób, by czytelnik **innego** artykułu czy **innej** książki napotkawszy na niezrozumiały termin – mógł zajrzeć do *Leksykonu*, otrzymać **jak najszybciej** wytłumaczenie dręczącej go kwestii i żeby mógł zaraz powrócić do czytania tego, co go naprawdę interesuje. Dlatego poszczególne hasła opracowano tak, by żadne z nich nie zajmowało w większości przypadków więcej niż jednej stronicy – włączając w to zarówno tekst, jak i obowiązkowy rysunek. *Leksykon* objaśnia więc poszczególne znajdujące się w nim hasła, terminy i pojęcia bardzo przystępnie, ale też bardzo krótko. Przy pisaniu *Leksykonu* zdarzało się jednak często, że starając się wyjaśnić maksymalnie zwarcie jedno pojęcie –- musieliśmy w objaśnieniu użyć innych specjalistycznych terminów. Bardzo często takie terminy będą dla czytelnika zrozumiałe intuicyjnie, ale pisząc *Leksykon* zakładaliśmy, że **nic** nie jest oczywiste i dla każdego terminu jest zbudowana osobna strona, zawierająca jego objaśnienie. W tradycyjnej książce konieczność zaglądania (w razie potrzeby) do dodatkowych objaśnień byłaby bardzo uciążliwa i kłopotliwa. Na szczęście ta książka wydana jest w formie elektronicznej, co powoduje, że przy jej studiowaniu można korzystać z łączników hipertekstowych. Dzięki tym łącznikom można w każdej chwili kliknąć niezrozumiałe słowo i natychmiast dostać kolejne krótkie i zwarte objaśnienie – jeśli oczywiście budzące wątpliwości słowo ma w *Leksykonie* swoją reprezentację. Ale to ostatnie łatwo jest rozpoznać, bo słowa zawierające łączniki hipertekstowe są dyskretnie wyróżnione w tekście.

Dzięki małej objętości i popularnej formie informacje zawarte w *Leksykonie* można łatwo pozyskać i przyswoić, uzyskując w ten sposób swoistą „przepustkę" do czytania innych książek i artykułów mówiących o **zastosowaniach** sieci neuronowych.

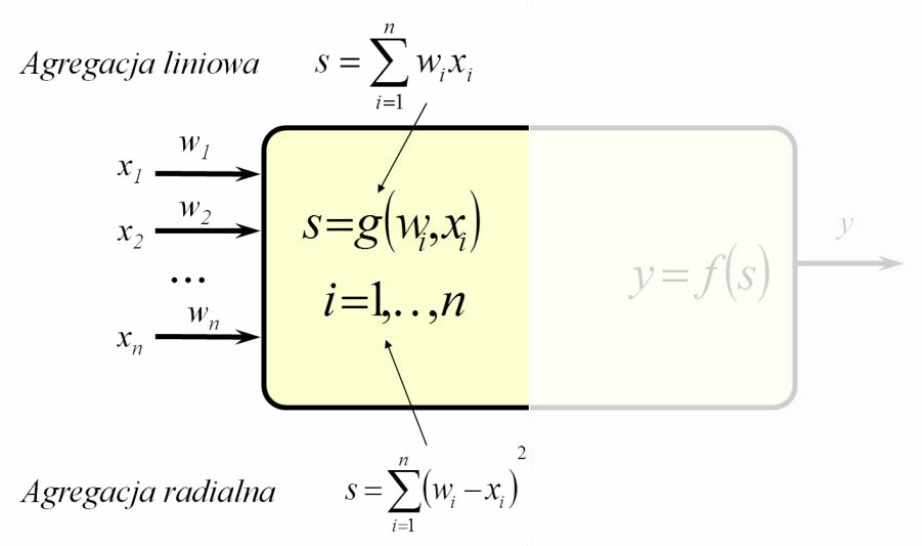
Na koniec tego wstępu powtórzmy jeszcze raz to, co zostało napisane w monografii [9], w której zawarty był *Dodatek* zatytułowany *Kompendium sieci neuronowych*, będący swoistym prototypem tego *Leksykonu*:

> „Zasób wiedzy zgromadzonej w **Leksykonie** jest naprawdę minimalny. Jest on jak żelazna racja żywności w szalupie albo jak samochodowa apteczka na wycieczce. Taki zestaw-minimum może pomóc w nagłej potrzebie, ale nie da się przy jego pomocy nasycić ani wyleczyć. Dlatego korzystając z **Leksykonu** Czytelnik musi mieć świadomość, że po bardziej obszerne wyjaśnienia czy po bardziej szczegółowe informacje będzie musiał sięgnąć do dalszych źródeł."

Na przykład do książek wymienionych w bibliografii w pozycjach od [1] do [7]. Chociaż nie są to książki wyłącznie najnowsze, do dziś nie straciły one aktualności, bo mimo ogromnego postępu w zakresie zastosowań sieci neuronowych, ich podstawy teoretyczne, zasady działania i ogólne właściwości – w gruncie rzeczy nie zmieniły się od lat 90. XX wieku!

Agregacja danych wejściowych

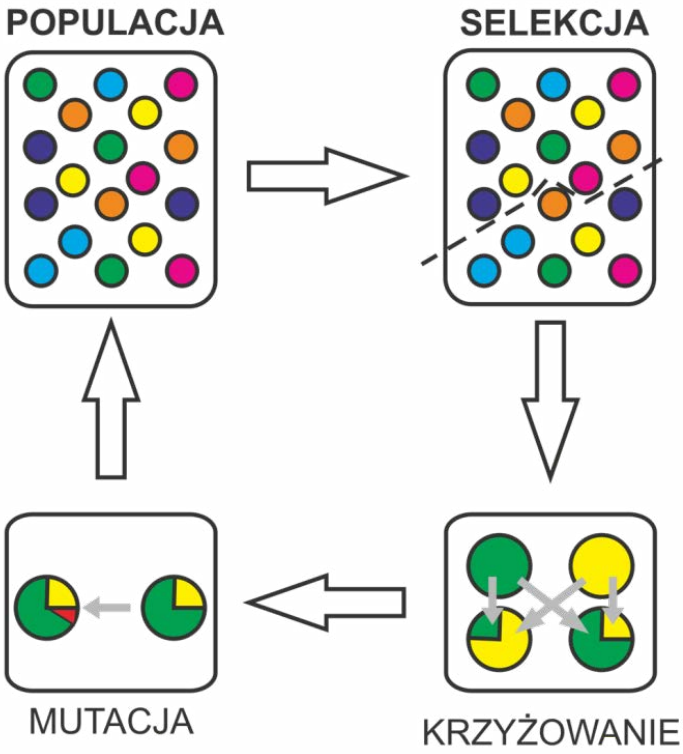
Jest to pierwsza czynność, jaką wykonuje **neuron**. Ponieważ **neuron** ma zwykle wiele **wejść** i jedno **wyjście** – konieczne jest przekształcenie wielu **danych wejściowych** w jeden wypadkowy sygnał **sumarycznego pobudzenia**, który kształtuje potem **sygnał wyjściowy neuronu** za pośrednictwem wybranej **funkcji aktywacji**.



W **neuronach** stosuje się różne formuły agregacji danych wejściowych, najczęściej jednak stosowana jest pokazana na rysunku agregacja liniowa (u góry) albo agregacja radialna (u dołu rysunku).

Algorytm genetyczny

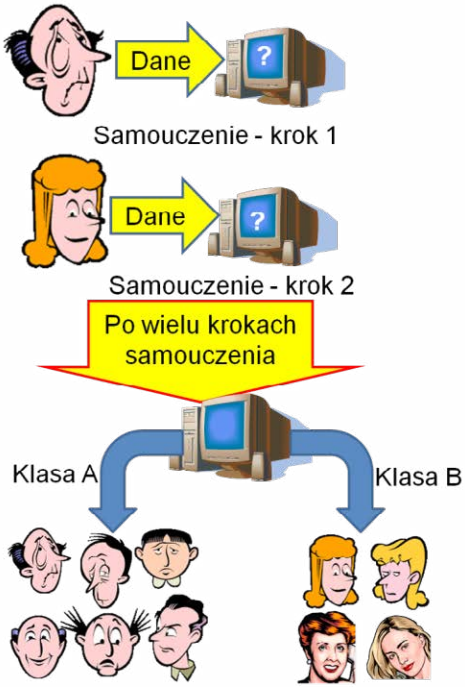
Metoda optymalizacji używana między innymi do wyboru najlepszej **struktury sieci neuronowej**. Schemat postępowania przy korzystaniu z algorytmu genetycznego jest przedstawiony na rysunku. Na początku wybierana jest pewna liczba początkowych rozwiązań (na przykład **struktur sieci** w ramach rozwiązania określanego jako **genetyczna sieć neuronowa**) tworzących POPULACJĘ. Rozwiązania te konfrontuje się z zadaniem, które należy rozwiązać (wszystkie sieci podlegają **uczeniu**) a następnie przeprowadzana jest SELEKCJA (w przypadku **sieci neuronowej** kryterium selekcji jest **egzamin**). Te rozwiązania, które przeszły pozytywnie selekcję (miały lepsze wyniki na **egzaminie**), poddawane są operacji KRZYŻOWANIA (wybrane losowo pary rozwiązań wymieniają losowo wybrane części swoich parametrów), tworząc rozwiązania *potomne*. Po wprowadzeniu dodatkowych losowych korekt nazywanych MUTACJAMI, rozwiązania *potomne* zastępują w POPULACJI rozwiązania *rodzicielskie* – i cykl się powtarza.

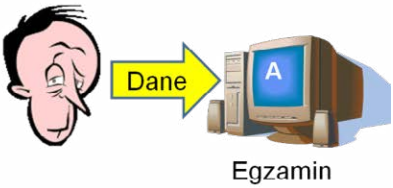


Algorytm samouczenia

Jest to metoda całkowicie samodzielnego dostosowywania się **sieci neuronowej** do analizy danych pojawiających się na jej **wejściu** bez żadnego komentarza ani wyjaśnienia. Sieć samoucząca potrafi takie dane automatycznie porządkować, dzielić na grupy, identyfikować i kategoryzować. Należy podkreślić, że w odróżnieniu od uczenia, kiedy sieć wyłącznie pozyskuje wiedzę pochodzącą od nauczyciela – przy samouczeniu musi ona tę wiedzę niejako sama odkrywać, co stwarza zupełnie nowe możliwości zastosowań.

Podobnie jako przy **algorytmie uczenia** możliwości zastosowania rozważanej klasy sieci wynikają z tego, że wytrenowana sieć neuronowa zdobytą wiedzę potrafi uogólniać (**generalizacja**). Powoduje to, że samodzielnie stworzone przez sieć kryteria kategoryzacji i klasyfikacji **danych wejściowych** mogą być wykorzystane także do tego, żeby klasyfikować bądź kategoryzować nowe dane, których sieć nie miała możności poznać w procesie samouczenia. Oczywiście te dane, na które sieć uogólnia swoją wiedzę, powinny należeć do tej samej klasy problemów, jak dane wykorzystywane podczas samouczenia.



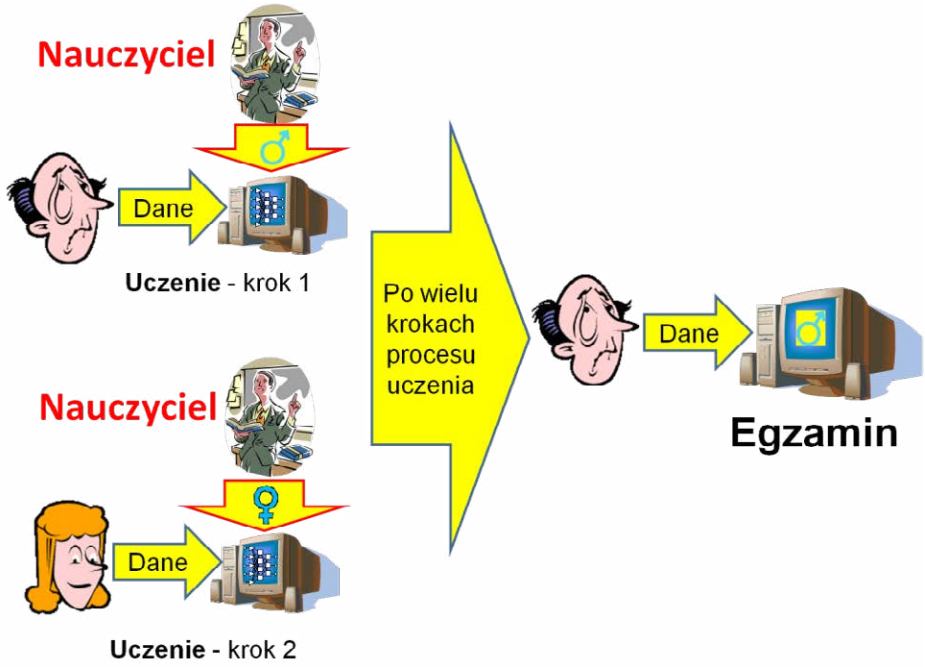


Wadą samouczenia jest to, że sieć wprawdzie całkiem sama zdobywa wiedzę (czy może nawet samodzielnie ją tworzy) – ale użytkownik nie ma żadnego wpływu na to, w jakiej formie ta wiedza jest przedstawiona na **wyjściu** sieci. W związku z tym interpretacja wyników samouczenia oraz rozwiązań dostarczanych przez wytrenowaną samouczącą się sieć w trakcie **egzaminu** wymaga pewnego wysiłku. Użytkownik musi najpierw zrozumieć, co sieć sygnalizuje i w jaki sposób – a dopiero potem może odnosić korzyści z samouczącej się sieci.

Najbardziej znanym przykładem sieci samouczącej się jest **sieć Kohonena**.

Algorytm uczenia

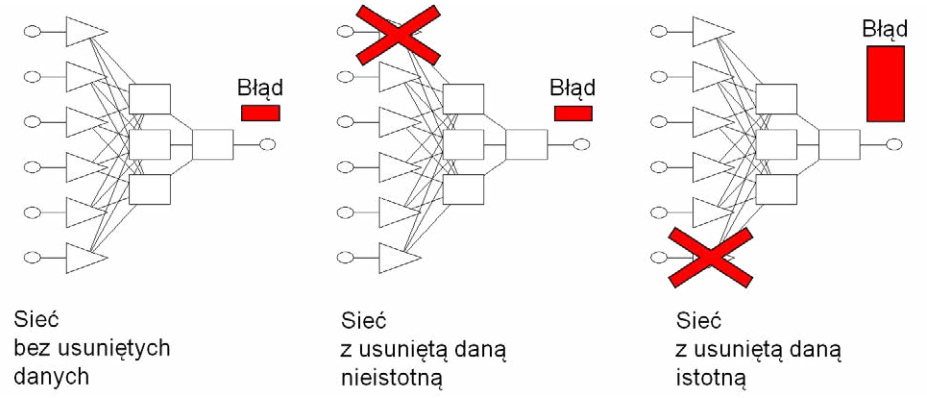
Jest to metoda dostosowywania **sieci neuronowej**, będącej potencjalnie narzędziem możliwym do wykorzystania w kontekście różnych zastosowań, do rozwiązywania określonego typu zadań, wyspecyfikowanych poprzez przykłady rozwiązań zawarte w **zbiorze uczącym**. Działanie algorytmu uczenia polega na pokazywaniu kolejnych **przypadków uczących** wraz z informacją podawaną przez **nauczyciela**, dotyczącą wymaganej poprawnej **odpowiedzi** sieci (tak zwana **odpowiedź wzorcowa**). Szczegóły algorytmu uczenia (a właściwie wielu różnych algorytmów, bo jest ich znanych obecnie kilkadziesiąt, jeśli nie kilkaset) są zbyt złożone na to, żeby je tu można było przedstawić. Ogólna idea **procesu uczenia** polega na minimalizacji **funkcji błędu**. Podczas działania algorytmu uczenia dochodzi do iteracyjnego modyfikowania **wag** w **sieci neuronowej**. Kryterium zatrzymania algorytmu związane jest z wykorzystaniem **zbioru walidacyjnego** sygnalizującego moment, kiedy sieć zaczyna tracić zdolność **generalizacji** wyników uczenia.



Po wykonaniu wielu kroków uczenia sieć jest gotowa do **egzaminu**, który sprawdza jej wiedzę i zdolność do jej **generalizacji**.

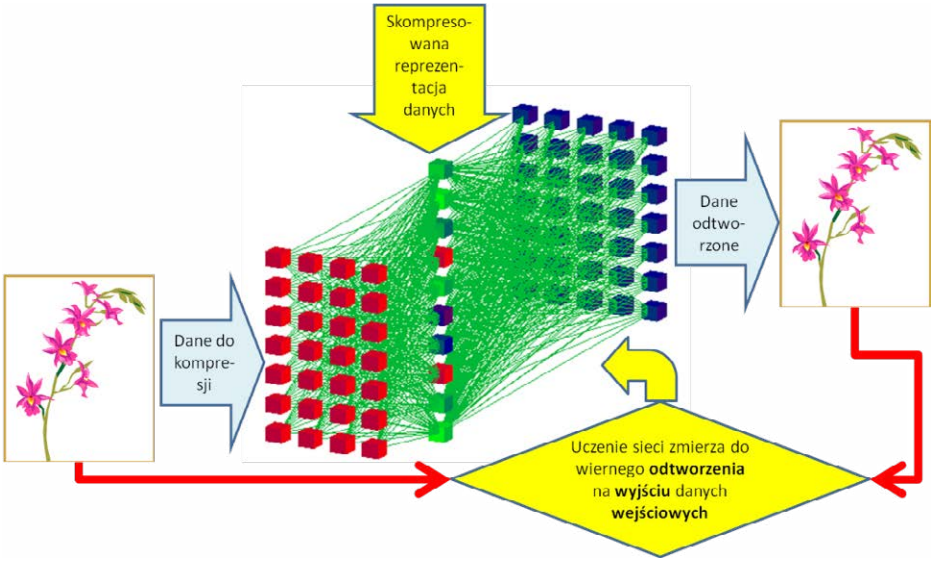
Analiza wrażliwości

Jeżeli w rozwiązywanym przez **sieć neuronową** problemie jest niewiele nieistotnych **danych wejściowych** a istnieje potrzeba zmniejszenia ich ilości, to można je wyeliminować na podstawie analizy wrażliwości. Analiza ta prowadzona jest po **uczeniu sieci neuronowej** i wykazuje, które **dane wejściowe** są najbardziej istotne. Poznajemy to poprzez analizę wzrostu **błędu** w przypadku eliminacji z **danych wejściowych** poszczególnych zmiennych. Usuwając z danych wejściowych pojedyncze zmienne uznane za nieistotne należy za każdym razem przeprowadzać **proces uczenia sieci** od początku.



Autoasocjacyjna sieć neuronowa

Jako autoasocjacyjna określana jest **sieć neuronowa**, w której **warstwa wejściowa** ma ten sam rozmiar (tę samą liczbę **neuronów**), co **warstwa wyjściowa**. Ponadto uczenie takiej sieci zmierza do wiernego odtworzenia na **wyjściu** przyjętych **danych wejściowych**. Uzasadnieniem dla stosowania takiej struktury sieci jest fakt, że pomiędzy **warstwą wejściową** sieci a **warstwą wyjściową** jest zwykle przynajmniej jedna **warstwa ukryta**, zawierająca znacznie mniej **neuronów** niż **warstwy wejściowa** i **wyjściowa**. W tej pośredniej warstwie wytwarzana jest skompresowana reprezentacja danych, zaś struktura sieci pomiędzy warstwą wejściową a wspomnianą warstwą pośrednią staje się narzędziem do kompresji danych. Z kolei ta część sieci, która rozciąga się od warstwy pośredniej do warstwy wyjściowej staje się narzędziem do dekompresji.



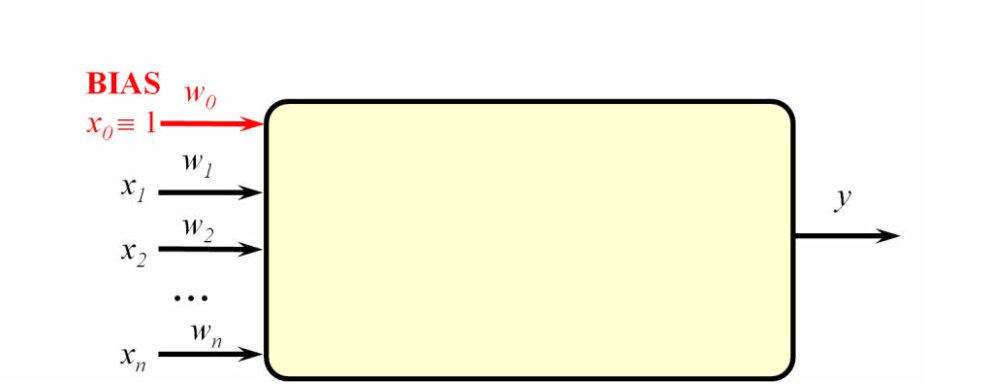
Sieci autoasocjacyjne bywają też wykorzystywane do realizacji nieliniowej **transformacji PCA**.

Backpropagation

Patrz hasło: **Wsteczna propagacja błędów**.

BIAS – wyraz wolny

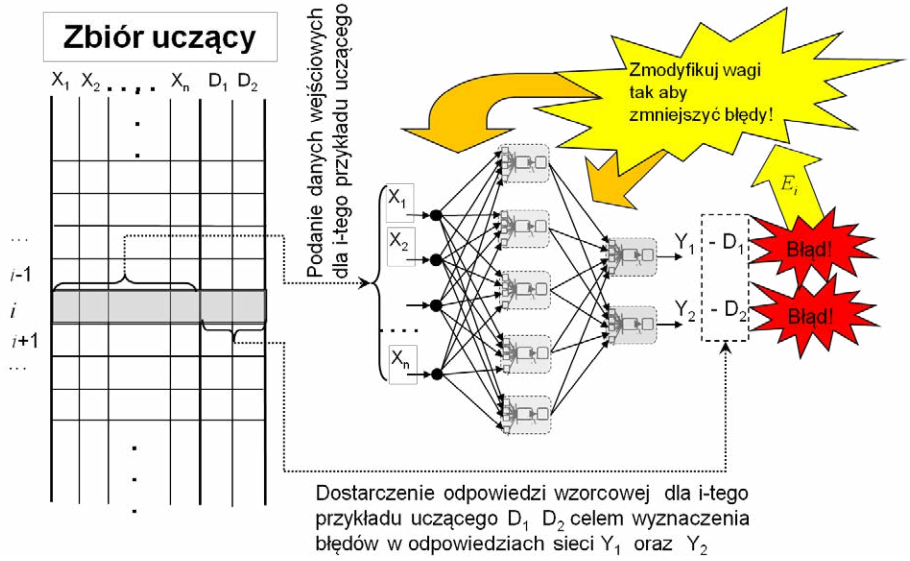
Większość **wag** w sieci neuronowej ma związek z **sygnałami** pojawiającymi się na **wejściach** poszczególnych neuronów. Sygnały te pochodzą albo od **danych wejściowych** podawanych do sieci jako opis zadania, które należy rozwiązać, albo z **wyjść neuronów** należących do wcześniejszej **warstwy sieci**. Czasami przy **uczeniu sieci** przydatne okazują się **wagi**, które nie są związane z żadnym z **sygnałów**. Takie wyrazy wolne w równaniach opisujących **neurony** oraz całe **sieci** pozwalają lepiej reprezentować zadania, które **sieć** powinna rozwiązywać. Dla ujednolicenia opisu **neuronów** korzystających z takich *wyrazów wolnych* i tych, które z nich nie korzystają, wprowadza się często do **struktury sieci neuronowej** generatory sztucznego pseudo-sygnału, określanego mianem BIAS. Sygnał ten ma z definicji zawsze wartość +1 i jest podawany na *dodatkowe* **wejście** neuronu. **Waga** związana z tym sygnałem podlega jednak **procesowi uczenia** podobnie jak wszystkie inne **wagi**, z tym że w odpowiednich formułach **algorytmu uczenia** w miejscu rzeczywistych sygnałów – występuje BIAS.



Błąd

Dla **neuronów** należących do **warstwy wyjściowej** jest to miara rozbieżności pomiędzy wartościami **danych wyjściowych** na **wyjściach** tych **neuronów** a wartościami **odpowiedzi wzorcowych** zawartych w **zbiorze uczącym**. Dla neuronów w **warstwach ukrytych** błąd musi być wyznaczany poprzez **wsteczną propagację**.

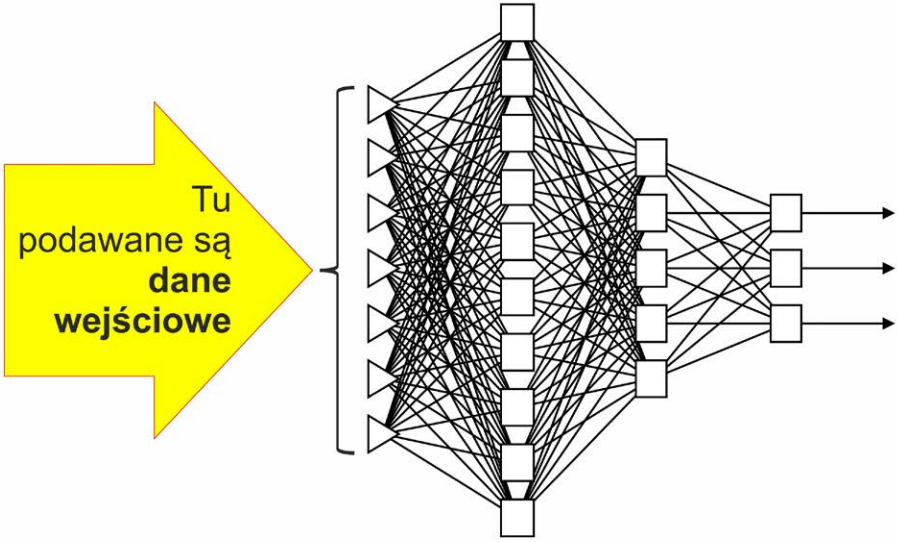
Błąd jest bardzo ważnym pojęciem w **procesie uczenia** sieci, ponieważ **wagi** w poszczególnych neuronach sieci modyfikuje się w taki sposób, by minimalizować popełniany przez sieć błąd.



Proces uczenia sieci neuronowej można w związku z tym traktować jako proces minimalizacji **funkcji błędu**.

Dane wejściowe

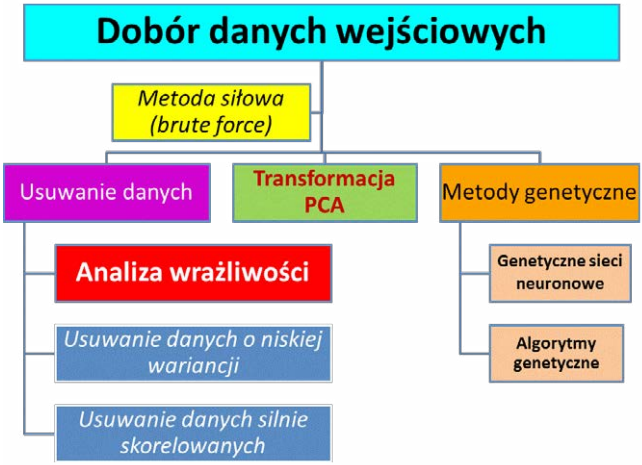
Są to informacje w formie **sygnałów** podawanych do wejść neuronów należących do **warstwy wejściowej sieci**. Uważa się, że dane wejściowe dostarczają wszystkie informacje niezbędne do tego, żeby sieć mogła rozwiązać postawiony problem. Dane wejściowe powinny być odpowiednio przygotowane, żeby mogły prawidłowo działać w sieci neuronowej.



Dobór danych wejściowych

W sytuacji gdy dysponujemy niewielką liczbą **przypadków uczących** zalecane jest przeprowadzenie selekcji **danych wejściowych** w celu zmniejszenia liczby **wag** koniecznych do wyznaczenia w **procesie uczenia**. Jest kilka metod, których można użyć:

1. *Metoda siłowa (brute force).* Tworzone są **sieci** wykorzystujące wszystkie możliwe podzbiory zbioru **danych wejściowych**. Jeżeli zostanie zastosowana taka sama metoda **uczenia** do wszystkich tych **sieci**, to wybór najlepszego zestawu **danych wejściowych** może być podyktowany najmniejszą wartością **błędu** uzyskanego po uczeniu na **zbiorze walidacyjnym**.
2. *Usuwanie danych silnie skorelowanych.* Metoda polega na wyznaczeniu korelacji par danych wejściowych i usunięcie jednej ze zmiennych z tych par, dla których współczynnik korelacji jest większy niż przyjęty próg, np. 0,9.
3. *Usuwanie danych o niskiej wariancji.* Jeżeli wariancja zmiennej wejściowej jest niska, to jej wartość informacyjna jest najprawdopodobniej niewielka i można ją pominąć.
4. **Analiza wrażliwości** – omówiona osobno.
5. **Transformacja PCA** – omówiona osobno.
6. **Algorytmy genetyczne** – omówione osobno.
7. **Genetyczne sieci neuronowe** – omówione osobno.

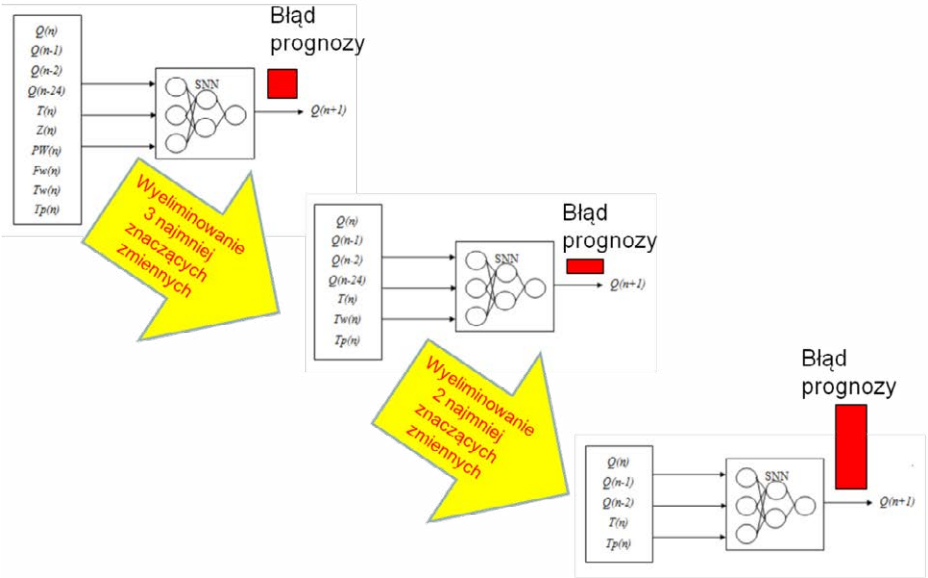


Patrz także hasło: **Dobór danych wejściowych – przykład**.

Dobór danych wejściowych – przykład

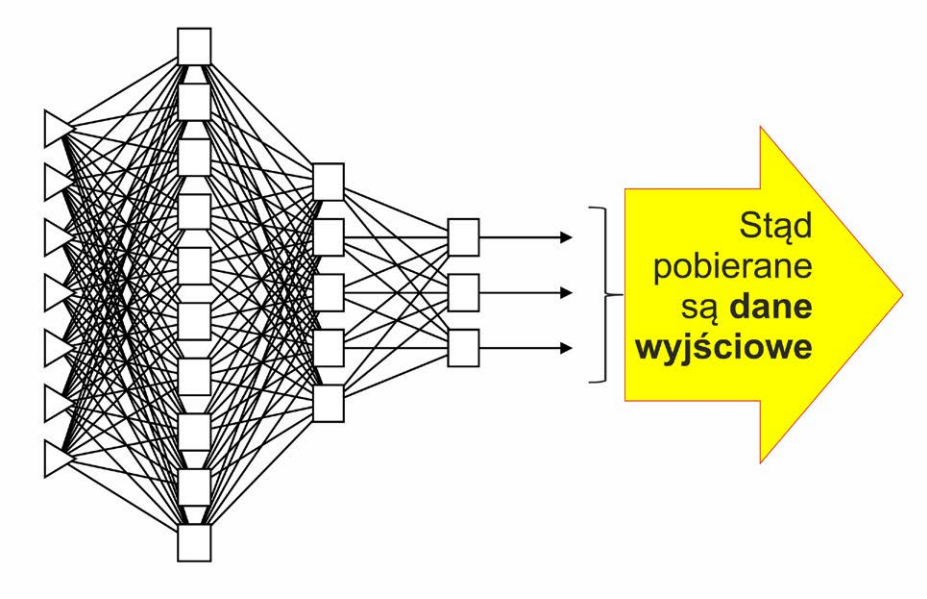
Przedstawiony przykład ilustruje **dobór danych wejściowych**, przeprowadzany dla konkretnego zadania, metodą wykorzystującą **analizę wrażliwości**.

W zadaniu, którego celem było prognozowanie zapotrzebowania na energię cieplną dla Krakowa, zastosowano początkowo **sieć** uwzględniającą 10 **danych wejściowych**. Otrzymano **błąd** prognozy 7,2%. Przeprowadzono **analizę wrażliwości** i stwierdzono, że trzy zmienne wejściowe można wyeliminować. Zmodyfikowana sieć (o 7 wejściach) po nauczeniu miała błąd prognozy 1,8%. Powtórzono procedurę, eliminując kolejne dwie najmniej wartościowe dane wejściowe. Sieć po tej kolejnej redukcji (5-wejściowa) miała jednak aż 22,3% błędu. Jak widać, przy redukcji danych wejściowych trzeba zachować umiar.



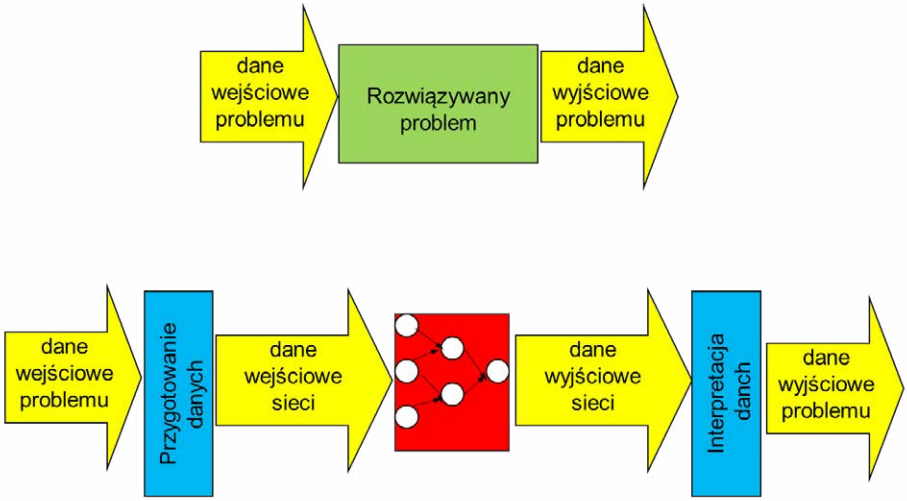
Dane wyjściowe

Są to informacje w formie sygnałów pojawiające się na **wyjściach** neuronów **wyjściowej warstwy** sieci. Informacje te podają rozwiązanie postawionego problemu. Zwykle, żeby wykorzystać **dane wyjściowe** jako rozwiązanie problemu, trzeba je odpowiednio zinterpretować, bo same wartości pojawiające się na wyjściach neuronów należących do wyjściowej warstwy sieci nie zawsze są same z siebie wystarczająco zrozumiałe. Sposób interpretacji danych wyjściowych zdefiniowany jest najczęściej w **zbiorze uczącym**. Zawarte w tym zbiorze **odpowiedzi wzorcowe** pokazują, jak należy rozumieć poszczególne dane wyjściowe w kontekście zadań, które sieć ma rozwiązywać po nauczeniu.



Dostosowanie rozwiązywanego problemu do użycia sieci neuronowej

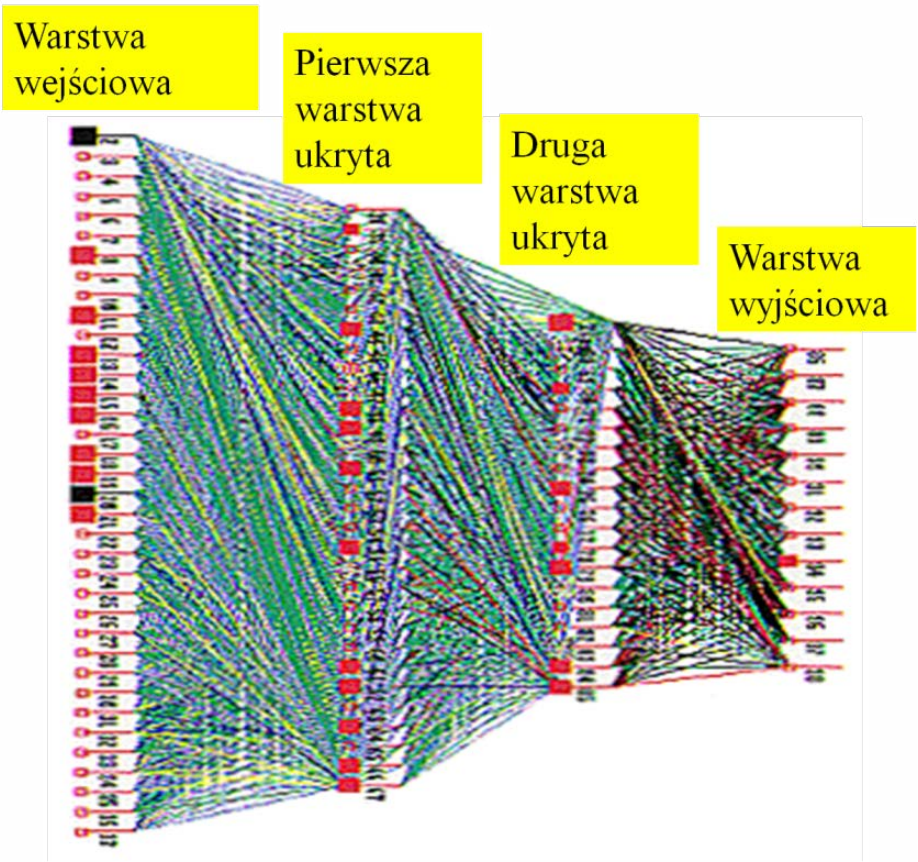
Przy pomocy **sieci neuronowej** można rozwiązywać różne problemy, zwykle jednak zanim użyje się sieci, trzeba rozwiązywany problem dostosować do jej użycia. Dostosowanie to polega na odpowiednim **przygotowaniu ilościowych danych wejściowych dla sieci neuronowej** oraz (jeśli zachodzi potrzeba) **przygotowaniu jakościowych danych wejściowych dla sieci neuronowej**, a także na ustaleniu odpowiedniej **interpretacji jakościowych danych wyjściowych z sieci neuronowej** (zmienne wyjściowe o charakterze ilościowym na ogół specjalnej interpretacji nie wymagają).



Stawianie sieci zadania bywa bardzo trudne bez wskazanego tu dostosowania (zarówno przy procesie **uczenia**, jak i w trakcie **egzaminu**) oraz trudno jest wykorzystywać **dane wyjściowe** uzyskiwane z sieci do generacji rozwiązań postawionego problemu.

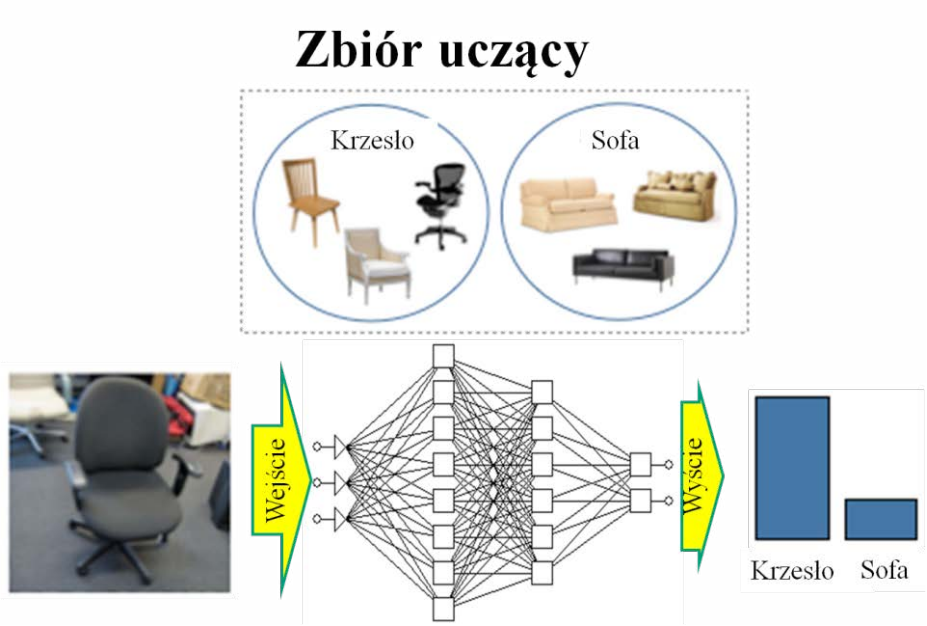
Duża sieć neuronowa

Schematy sieci neuronowych przedstawianych w poszczególnych hasłach tego *Leksykonu* są na ogół przedstawiane w taki sposób, że zawierają niewiele **neuronów** w poszczególnych **warstwach** i w efekcie niewiele **połączeń** (porównaj na przykład rysunek w haśle **struktura sieci**). Tymczasem prawdziwe sieci neuronowe, używane przez Autorów różnych artykułów naukowych, zawierają mnóstwo neuronów ukrytych oraz najczęściej przynajmniej kilka wyjść. Przykładowy schemat takiej dużej sieci neuronowej przedstawia poniższy rysunek, na którym widać, że w dużej sieci trudno jest prześledzić **strukturę** i niewygodnie jest wnioskować na temat jej działania – stąd celowość przedstawiania na schematach sieci uproszczonych.



Egzamin

Potoczna nazwa procesu testowania efektów **uczenia** lub **samouczenia** sieci neuronowej. Podczas egzaminu powinno się używać zadań (**przypadków uczących**), których prawidłowe rozwiązania są znane, ale które nie były wcześniej używane ani w charakterze elementów **zbioru uczącego**, ani elementów **zbioru walidacyjnego**. Często dla potrzeb egzaminu tworzy się specjalny **zbiór testowy**. Zasadniczym celem egzaminu jest sprawdzenie zdolności sieci do **generalizacji** zdobytej wiedzy. Przykład **egzaminu sieci** neuronowej przedstawia rysunek poniżej.

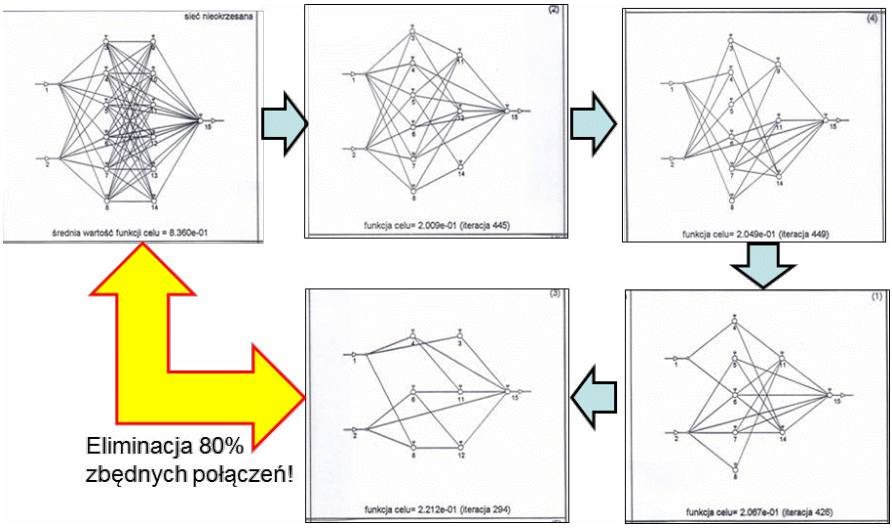


Odpowiedzi sieci podczas egzaminu mogą nie być tak kategoryczne jak podczas uczenia, ale jeśli sygnał optujący za prawidłowym rozwiązaniem jest silniejszy niż sygnał odpowiedzi fałszywej – egzamin można uznać za zaliczony.

Eliminacja zbędnych połączeń

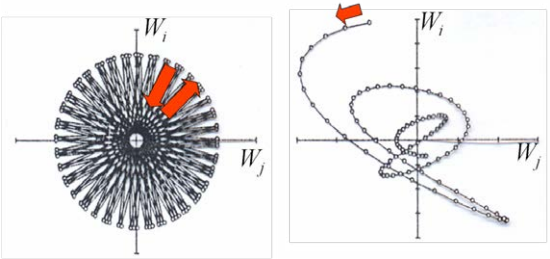
Stosowana powszechnie przy tworzeniu warstwowych sieci neuronowych zasada połączeń „**każdy z każdym**" skutkuje tym, że w **sieci neuronowej** przed rozpoczęciem procesu **uczenia** wiele połączeń jest niepotrzebnych. Z góry nie wiadomo, które połączenia są zbędne, ale można je eliminować już po nauczeniu sieci, ponieważ jako zbędne można zakwalifikować te wszystkie połączenia, dla których zakończony sukcesem proces **uczenia** ustalił zerowe lub bliskie zera wartości **wag**. Usuwanie zbędnych połączeń wiąże się często z usuwaniem z sieci niepotrzebnych neuronów – takich, których **wyjścia** na skutek eliminacji połączeń przestały być wykorzystywane przez inne neurony sieci. Sieć z usuniętą częścią połączeń bywa douczana, w wyniku czego możliwa staje się eliminacja kolejnych zbędnych połączeń.

Usuwanie zbędnych połączeń bywa często określane angielskich słowem *pruning*, oznaczającym oryginalnie przycinanie roślin (drzew owocowych, winorośli, żywopłotów itp.) w ogrodnictwie. Typowy *pruning* w sieci neuronowej przedstawia rysunek. Autorzy wiedzą o tym, że opisy na schematach kolejnych wersji sieci, w której dokonywano redukcji połączeń, są nieczytelne. Nie ma to jednak znaczenia, bo istotny jest tylko proces upraszczania struktury sieci, który jest dobrze widocznie i łatwo interpretowalny. Napisy pozostawiono, żeby uwidocznić fakt, że rysunki pochodzą z rzeczywistych badań rzeczywistej sieci neuronowej, a nie zostały sporządzone jedynie na podstawie wyobrażeń badacza.



Epoka

Podczas **uczenia sieci neuronowej** trzeba wykonać bardzo wiele kroków **algorytmu uczenia** zanim **błąd** stanie się akceptowalnie mały. Tymczasem **zbiór uczący** zawiera zwykle ograniczoną liczbę **przypadków uczących**, w typowych przypadkach setki lub nawet tysiące razy mniej liczną niż liczba koniecznych kroków algorytmu uczenia. Z tego zestawienia wynika, że **zbiór uczący** musi być wykorzystywany w **procesie uczenia** wielokrotnie. Dla zaznaczenia tego faktu wprowadzono pojęcie epoki, rozumiejąc pod tym pojęciem jednorazowe użycie w **procesie uczenia** wszystkich **przypadków uczących** zawartych w **zbiorze uczącym**. Po wykonaniu wszystkich kroków należących do jednej epoki **algorytm uczący** dokonuje oceny zdolności sieci do **generalizacji** wyników uczenia przy wykorzystaniu **zbioru walidacyjnego**. Po stwierdzeniu, że zarówno **błąd** obliczany na **zbiorze uczącym**, jak i **błąd** wyznaczony dla **zbioru walidacyjnego** nadal jeszcze obiecująco maleją – **algorytm uczący** wykonuje następną epokę. W przeciwnym przypadku **proces uczenia** zostaje zatrzymany.



Gdyby w kolejnych epokach **przypadki uczące** pokazywać stale w tej samej kolejności – to istniałaby obawa, że **proces uczenia** może zmieniać **wagi** w kółko, powracając po każdym cyklu do punktu wyjścia. Przedstawia to rysunek, na którym po lewej stronie pokazano właśnie taki „zapętlony" proces zmiany **wag**, nie prowadzący do nauczenia sieci nawet po bardzo długim **procesie uczenia**. Na rysunku pokazano cykliczne zmienianie się wartości dwóch wybranych **wag** (bo tylko to można pokazać na rysunku), ale podobny niekorzystny proces zachodzi także dla wszystkich innych **wag** w całej **sieci**.

Zapętleniu uczenia można zapobiec poprzez *randomizację* **zbioru uczącego**, to znaczy poprzez zmianę kolejności pokazywania poszczególnych **przypadków uczących** w kolejnych epokach. Wtedy proces zmiany **wag** w trakcie uczenia porządkuje się i wyraźnie widać, że zmierza do określonego celu, odpowiadającego optymalnemu zestawowi wag zapewniającemu rozwiązywanie stawianych **sieci** zadań z minimalnym błędem (co pokazano na rysunku po prawej stronie).

Feedforward

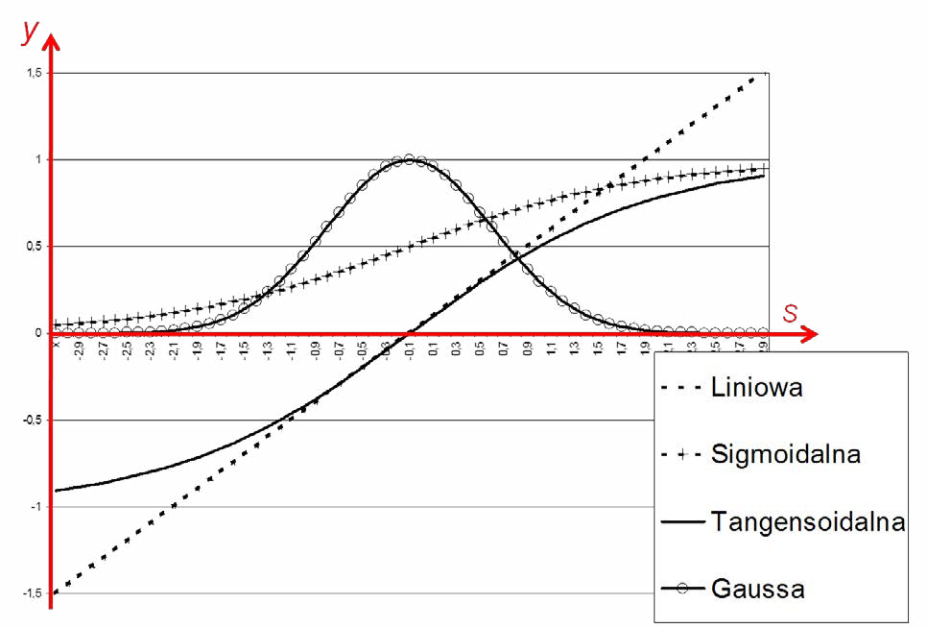
Patrz hasło **Jednokierunkowa sieć neuronowa**.

FNN (Fuzzy Neural Network)

Patrz hasło **sieć neuronowo-rozmyta**.

Funkcja aktywacji

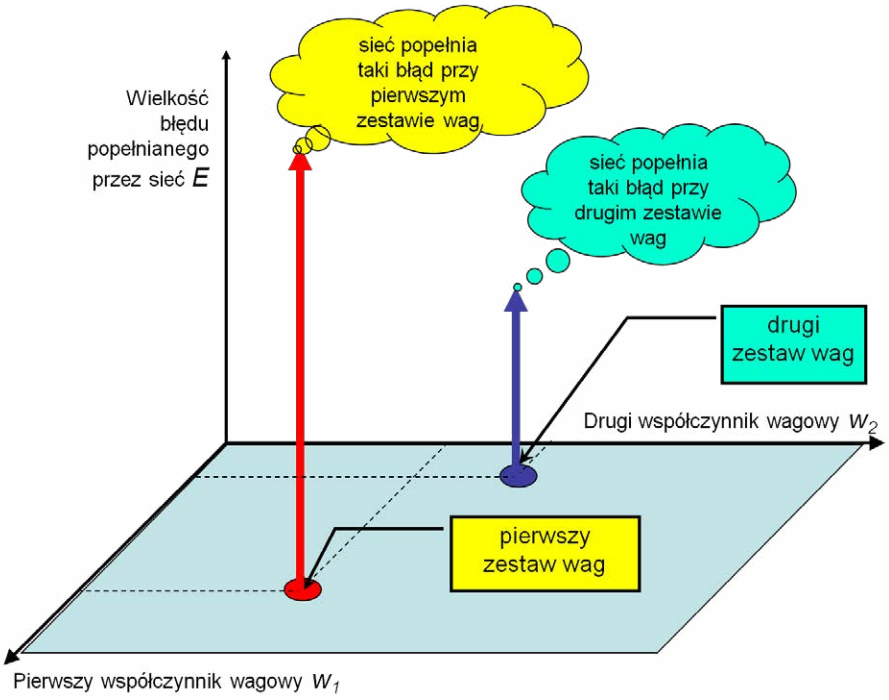
Po **agregacji danych wejściowych** z uwzględnieniem **wag** powstaje sygnał **sumarycznego pobudzenia**. Rola funkcji aktywacji polega na tym, że musi ona określić sposób obliczania wartości **sygnału wyjściowego neuronu** na podstawie wartości tego **sumarycznego pobudzenia**. W literaturze rozważano wiele różnych propozycji funkcji aktywacji, jednak do powszechnego użytku weszły właściwie cztery z nich: funkcja liniowa (**neuron liniowy**), funkcja sigmoidalna (**neuron sigmoidalny**), funkcja tangensoidalna (dokładnie jest to funkcja *tangens hiperboliczny*, ale skrótowo mówi się właśnie **neuron tangensoidalny**) oraz funkcja Gaussa (**neuron radialny**).



Osobnym (niepokazanym na rysunku) typem funkcji aktywacji jest funkcja hiperboliczna (wyliczana jako odwrotność argumentu) używana w **neuronach Kohonena**.

Funkcja błędu

Błąd popełniany przez sieć neuronową zależny jest od współczynników **wag** występujących w sieci i doskonalonych przez **algorytmy uczenia**. Jeśli wyobrazimy sobie (patrz rysunek), że w danym momencie **procesu uczenia** w sieci został ustalony pewien zestaw **wag** (nazwany na rysunku *pierwszym zestawem*) i jeśli przy tym zestawie **wag** przeprowadzimy **egzamin**, to uzyskamy pewną wartość **błędu**, przedstawioną na rysunku przy pomocy pionowej strzałki. Jeśli wartość zestawu **wag** się zmieni (na przykład w wyniku **uczenia**) i będziemy mieli do czynienia z *drugim zestawem* – to dla niego także można będzie wyznaczyć **błąd** i przedstawić go – jak na rysunku – przy pomocy niższej strzałki. Jeśli taką czynność wystawiania pionowych strzałek oznaczających wartości błędów wykonamy w każdym punkcie szarej płaszczyzny, reprezentującej na rysunku wszystkie możliwe zestawy **wag** – to wierzchołki strzałek wyznaczą pewną powierzchnię rozpiętą ponad szarą płaszczyzną. Właśnie ta powierzchnia to potrzebna do wielu celów (między innymi w opisie **procesu uczenia**) *funkcja błędu*.

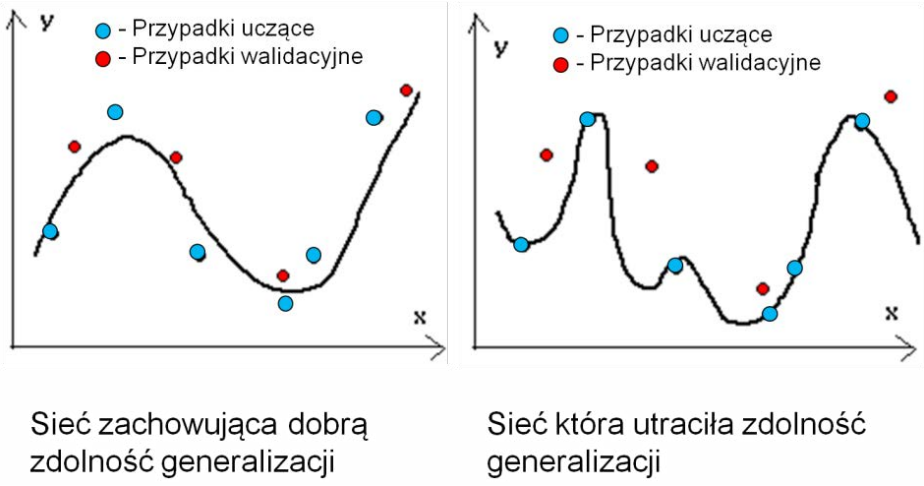


Do rysunku dodać trzeba pewien komentarz:

Żeby można było przedstawić na tym rysunku proces budowy funkcji błędu – trzeba było założyć, że funkcja ta zależy wyłącznie od dwóch współczynników wagowych rozpinających na rysunku szarą płaszczyznę. Jest to bardzo daleko idące uproszczenie, ponieważ w rzeczywistości funkcja błędu sieci neuronowej zależy od wszystkich wag występujących w całej sieci – a tych wag są setki, a czasem tysiące. Jednak funkcji zależnej od tysiąca zmiennych narysować się nie da, więc przedstawiono rysunek sytuacji uproszczonej (wyidealizowanej), żeby wytworzyć u Czytelnika ogólną intuicję. Ta intuicja pozwala też zrozumieć, co się dzieje w tej wielowymiarowej przestrzeni wag podczas rzeczywistego **procesu uczenia**.

Generalizacja

Proces uczenia opiera się zawsze na **zbiorze uczącym**, który zawiera **przypadki uczące**, czyli zadania wraz ze znanymi poprawnymi rozwiązaniami (**odpowiedzi wzorcowe**). Jednak sens użycia sieci neuronowej polega na tym, że musi ona (po nauczeniu) rozwiązywać zadania podobne do tych, na których była uczona – ale nie identyczne z nimi. Takie przeniesienie nabytej wiedzy na nowe przypadki nazywane jest generalizacją. Zagrożeniem dla **generalizacji** jest **przeuczenie**. Gdy sieć jest przeuczona – następuje nadmierne dopasowanie jej zachowania do drugorzędnych (nieistotnych) szczegółów konkretnych przypadków uczących – nie mających istotnego znaczenia z punktu widzenia istotnych cech rozwiązywanego zadania. Na rysunku pokazano to na przykładzie działania prostej sieci, której zadaniem jest odtworzenie przebiegu funkcji jednej zmiennej na podstawie zbioru punktów tworzących **zbiór uczący**.



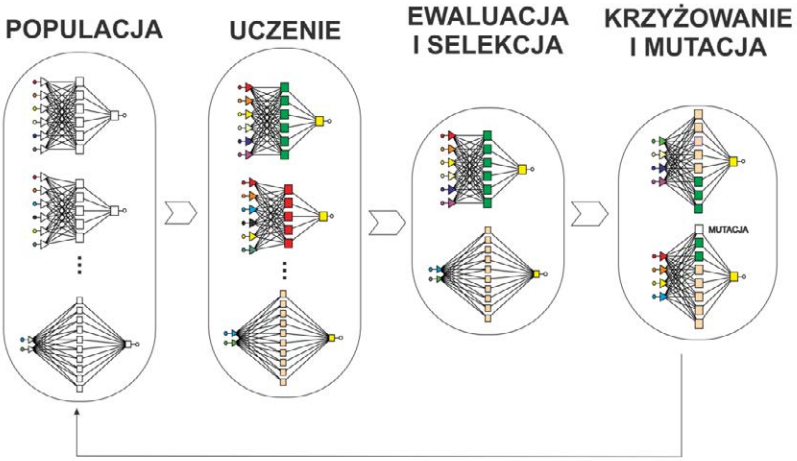
Do kontroli tego, czy sieć nie zatraciła zdolności do generalizacji, używa się **zbioru walidacyjnego**.

Genetyczna sieć neuronowa

Genetyczne Sieci Neuronowe (*Genetic Neural Networks* – GNN) są połączeniem koncepcji optymalizacji za pomocą **algorytmów genetycznych** oraz modelowania za pomocą **sieci neuronowych**. GNN mają tę przewagę nad klasycznymi **sieciami neuronowymi**, że w trakcie ich optymalizacji przeprowadzone zostaje nie tylko uczenie samych sieci, ale jeszcze dobór optymalnej **struktury sieci neuronowej** oraz **wektora wejściowego**.

GNN optymalizują się przy użyciu **algorytmów genetycznych**. Populacja tworzona jest przez **sieci neuronowe** charakteryzujące się zarówno różnymi **danymi wejściowymi** oraz/lub ilością **neuronów warstwy ukrytej**.

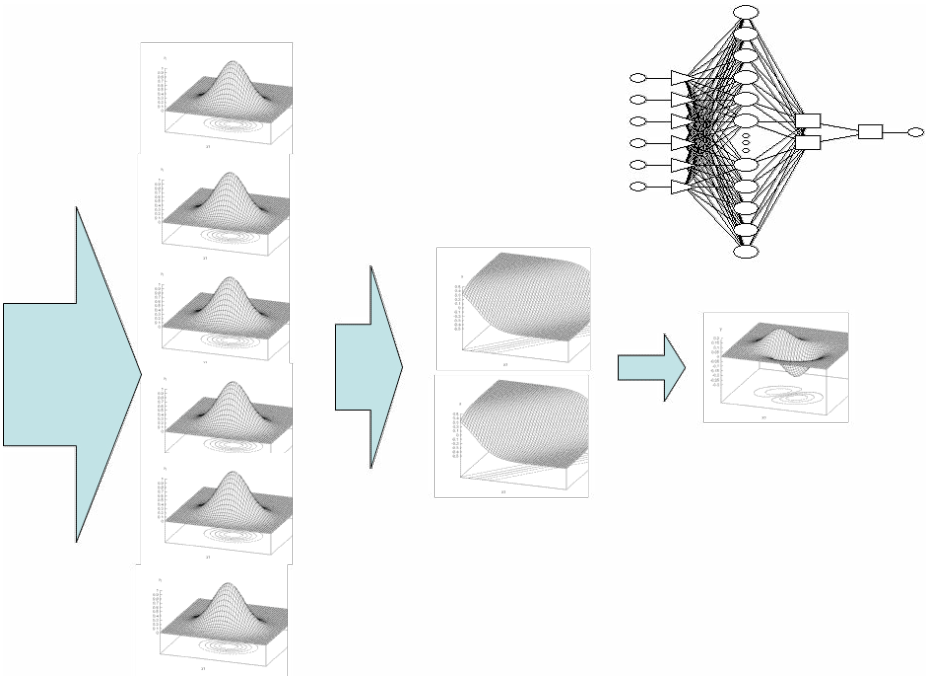
Podstawową zaletą stosowania GNN jest fakt, że funkcja przystosowania ocenia jakość nieliniowych modeli neuronowych i dlatego nie zachodzi niebezpieczeństwo wyeliminowania w czasie procedury optymalizacyjnej jakiegoś kluczowego nieliniowego związku. Ponadto metoda dostarcza zestawu dobrze działających **sieci**, które są w stanie rozwiązać dany problem często na różne, komplementarne sposoby. Wadą procedury GNN jest wysoki koszt obliczeniowy ze względu na iteracyjne trenowanie każdej sieci (np. 100 epok) w populacji (np. 100 modeli) w każdym pokoleniu ewolucji (np. 1000 pokoleń).



Patrz także **przebieg genetycznej optymalizacji sieci**.

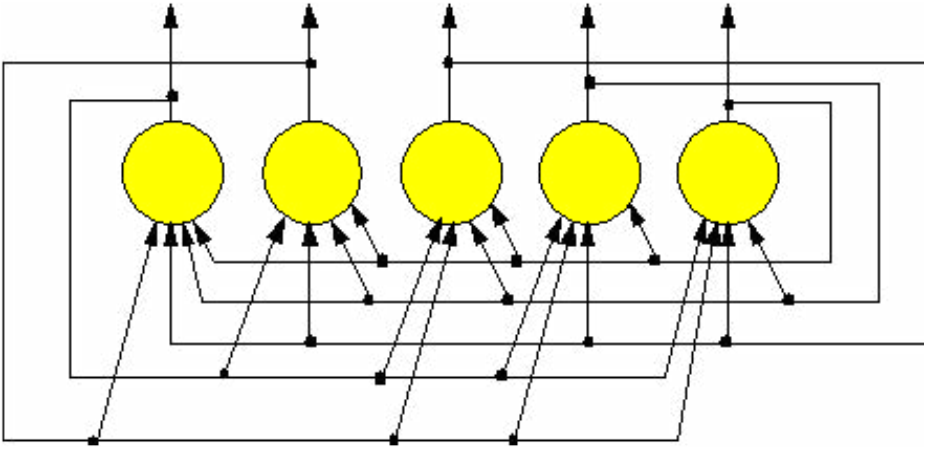
GRNN (*Generalized Regression Neural Networks*)

Szczegóły na temat budowy tej sieci: patrz hasło **sieć uogólnionej regresji**. W sieci GRNN wykorzystuje się połączenie właściwości neuronów **RBF** (z charakterystyką w formie funkcji Gaussa) oraz neuronów **MLP** (z charakterystyką sigmoidalną), co pozwala modelować wyjątkowo wyrafinowane zależności nieliniowe. Rysunek pokazuje, jak w sieci GRNN dochodzi do wytworzenia takiej złożonej nieliniowej zależności.



Hopfielda sieć

Jest to szczególny przypadek **sieci rekurencyjnej**. Sieć Hopfielda to sieć jednowarstwowa o regularnej budowie, składająca się z wielu **neuronów** połączonych **każdy z każdym**. Połączenie **wyjścia** neuronu *k* z **wejściem** neuronu *j* związane jest z **wagą** w_{kj}. Połączenia takie są zdefiniowane dla wszystkich *k* i *j*, co formuje ogromną liczbę sprzężeń zwrotnych zawartych w takiej sieci. Nie istnieją jednak **sprzężenia zwrotne** obejmujące ten sam neuron. Oznacza to, że **sygnał wyjściowy** danego neuronu nie trafia na jego wejście, a więc wartości **wag** w_{ii} są równe 0.



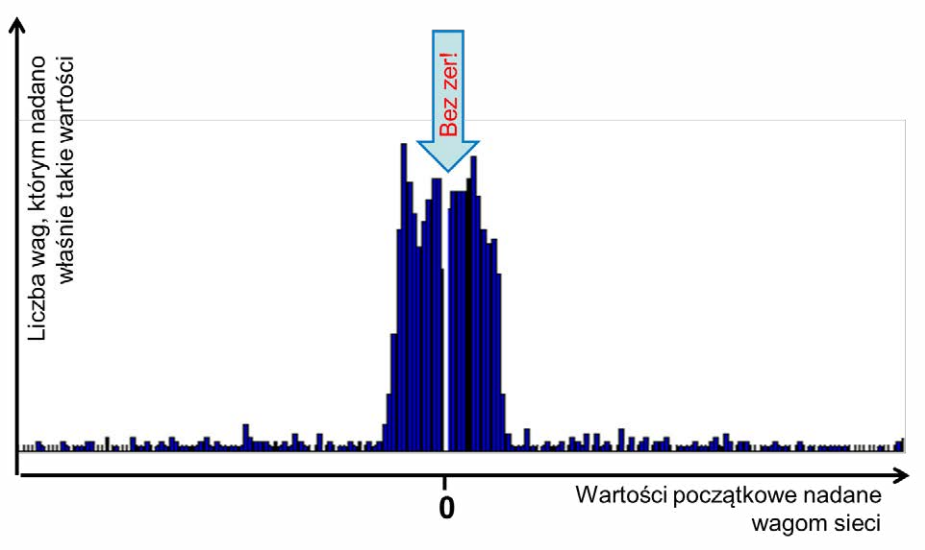
Ze względy na stabilność zachowania sieci zakłada się, że wagi w tej sieci są symetryczne, tzn. waga wkj łącząca neuron *k* z neuronem *j* jest równa wadze w_{jk} łączącej neuron *j* z neuronem *k*.

Sieć Hopfielda podczas **uczenia** modyfikuje swoje wagi w_{kj} w zależności od wartości **wektora wejściowego**. W trybie odtworzeniowym wagi nie ulegają modyfikacjom, natomiast **sygnał wejściowy** pobudza sieć, która poprzez sprzężenie zwrotne wielokrotnie przyjmuje na swoje wejście sygnał wyjściowy, aż do ustabilizowania odpowiedzi.

Sieci Hopfielda są wykorzystywane jako pamięci skojarzeniowe (zwłaszcza autoasocjacyjne) oraz jako narzędzia do znajdowania przybliżonych rozwiązań problemów optymalizacyjnych. Najbardziej znane osiągnięcie w tym zakresie dotyczy uzyskania za pomocą sieci Hopfielda przybliżonych (ale zadowalających z praktycznego punktu widzenia) rozwiązań znanego „problemu komiwojażera".

Inicjalizacja wag

Zanim **sieć** zacznie się **uczyć** z wykorzystaniem określonego **algorytmu uczenia**, podlegające uczeniu parametry sieci (najczęściej **wagi**) muszą mieć nadane wartości początkowe – żeby można je było w **procesie uczenia** poprawiać. To nadawanie wartości początkowych nazywa się inicjalizacją wag i polega na tym, że **wagom** nadaje się wartości losowe – przeważnie niewielkie pod względem wartości bezwzględnej, żeby ich zmiana w trakcie procesu uczenia nie nastręczała trudności. Na rysunku pokazano histogram **wag** po inicjalizacji przy pomocy algorytmu znanego jako *Nguyen-Widrow Randomization*.



Dla przypomnienia: Histogram jest to wykres pokazujący, jak często wystąpiły w sieci wagi o różnych wartościach. Widać, że większość **wag** ma wartości początkowe bliskie zera, chociaż bardzo duże dodatnie i ujemne wartości także się zdarzają. Samo zero jest na ogół wykluczane, bo **połączenie** z zerową **wagą** funkcjonuje tak, jak brak **połączenia**, więc sieć, w której na początku byłyby ustawione **wagi** zerowe, byłaby pozbawiona niektórych swoich **połączeń**.

Interpretacja jakościowych danych wyjściowych z sieci neuronowej

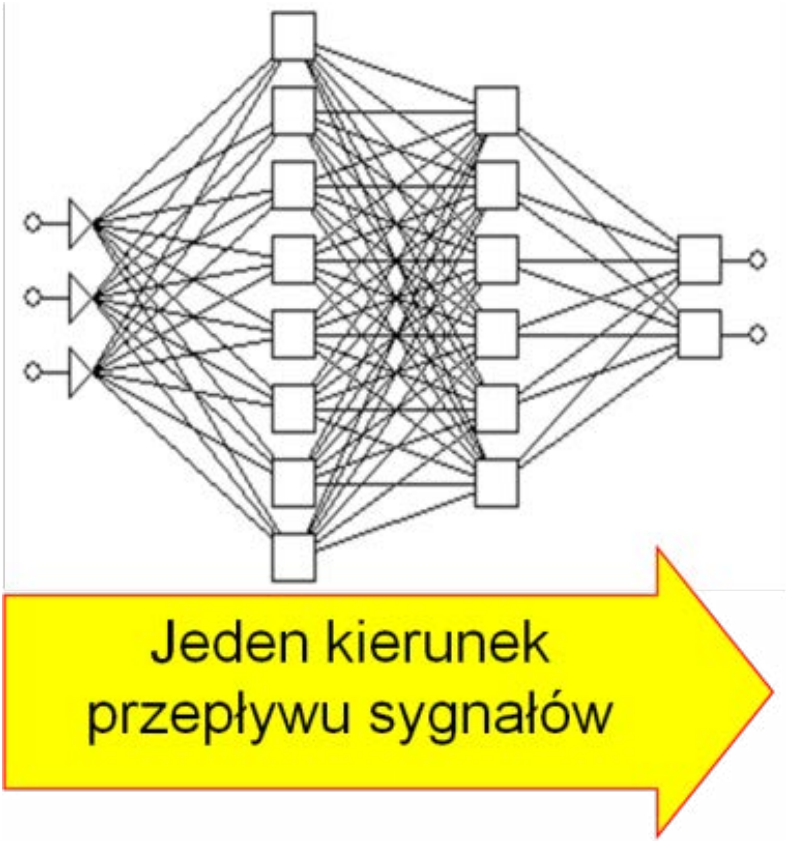
Jeśli w wyniku działania sieci powstają **dane wyjściowe** w postaci liczbowej – to właściwie stanowią one rozwiązanie podjętego problemu i poza ewentualnym przeskalowaniem nie wymagają żadnych dodatkowych zabiegów. Jeśli jednak sieć ma wyprodukować dane o charakterze jakościowym (to znaczy zasugerować decyzję lub wybrać jedną z możliwości), to stosujemy kodowanie *jeden-z-N* opisane w haśle **przygotowanie jakościowych danych wejściowych dla sieci neuronowej**. Zakładając, że sieć ma dostarczyć rozwiązania w postaci wskazania jednej z *N* możliwości, buduje się sieć mającą *N* neuronów wyjściowych. Każdej możliwości odpowiada jeden neuron w warstwie wyjściowej. Neurony wyjściowe produkują sygnały zgodne (najczęściej) z sigmoidalną **funkcją aktywacji**, czyli przyjmujące wartości od 0 do 1. Aby przyjąć decyzję, że **dana wyjściowa** jest równa *i-tej* możliwości, muszą być spełnione warunki:

- wartość **wyjścia** *i-tego* neuronu – wyższa od poziomu akceptacji,
- wartości **wyjść** pozostałych neuronów – niższe od poziomu odrzucenia.

O tym, co zrobić, jeśli podane wyżej warunki nie są spełnione, użytkownik sieci musi zdecydować, biorąc pod uwagę charakter rozwiązywanego zadania. Czasem można się zgodzić na wybór możliwości *i* nawet w przypadku, gdy wyjście z neuronu *i* nie przekracza progu akceptacji, ale grozi to błędną klasyfikacją **danych wejściowych**.

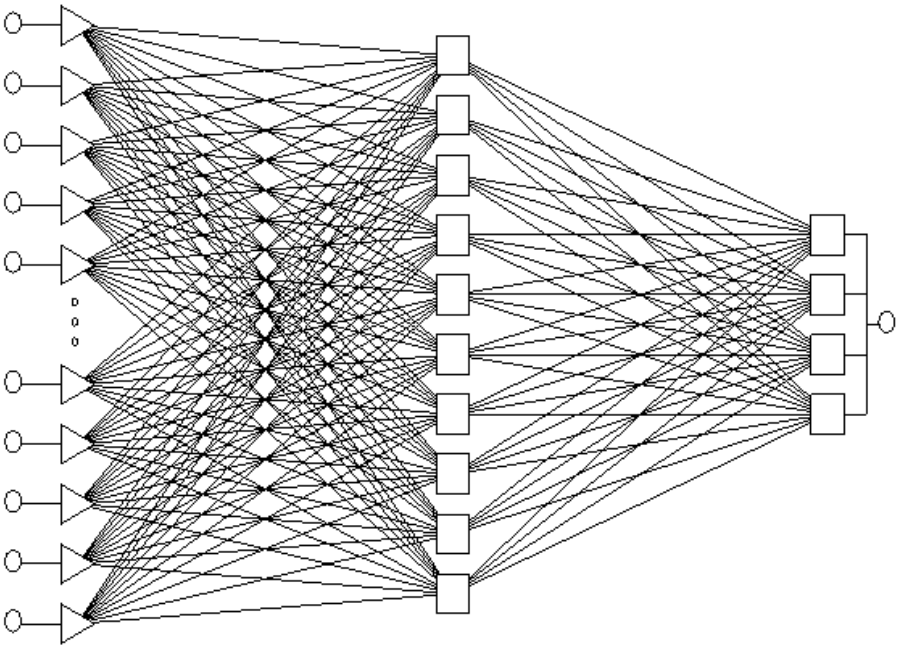
Jednokierunkowa sieć neuronowa

Sieci neuronowe budowane są zazwyczaj w taki sposób, że przepływ sygnałów odbywa się w nich wyłącznie w kierunku od **wejścia** (poprzez ewentualne **warstwy ukryte**) do **wyjścia**. Wykluczony jest przepływ sygnałów w drugą stronę, co powoduje, że sieci tego typu są przeciwstawiane **sieciom rekurencyjnym**. Sieci spełniające wyżej podany warunek nazywane są sieciami jednokierunkowymi albo sieciami typu **feedforward**. Sam przepływ sygnałów w jednym kierunku (od wejścia do wyjścia) nie przesądza jeszcze o rodzaju sieci i zasadzie jej działania, gdyż wśród jednokierunkowych sieci neuronowych wyróżnić można między innymi **wielowarstwowe perceptrony** (sieci **MLP**), **sieci radialne** (**RBF**), **sieci uogólnionej regresji** (**GRNN**), **probabilistyczne sieci neuronowe** (**PNN**) i inne. W praktyce autorzy najczęściej utożsamiają nazwę sieci jednokierunkowej z siecią typu **MLP**.



Każdy z każdym

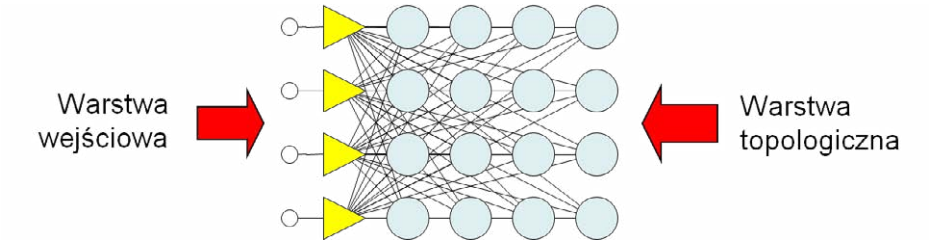
Zasada łączenia elementów sieci na zasadzie każdy z każdym wynika z faktu, że twórca sieci z góry nie wie, które połączenia okażą się potrzebne. Dlatego przy **połączeniach** międzywarstwowych z reguły zakłada się, że każdy **neuron** wcześniejszej **warstwy** jest połączony z każdym neuronem następnej **warstwy**. Dla uniknięcia nieporozumień wyjaśnijmy, że **warstwę** uważa się za wcześniejszą, jeśli jest położona bliżej **wejścia** sieci.



W trakcie procesu **uczenia sieci** dochodzi do tego, że **wagi** niektórych połączeń otrzymują wartości zerowe, co powoduje, że odpowiadające im połączenia są w istocie nieaktywne. W wyniku tego dochodzi do **redukcji połączeń** i sieć po nauczeniu może zawierać o wiele mniej połączeń i może również nie używać wielu neuronów, które można wtedy bez szkody usunąć z jej struktury.

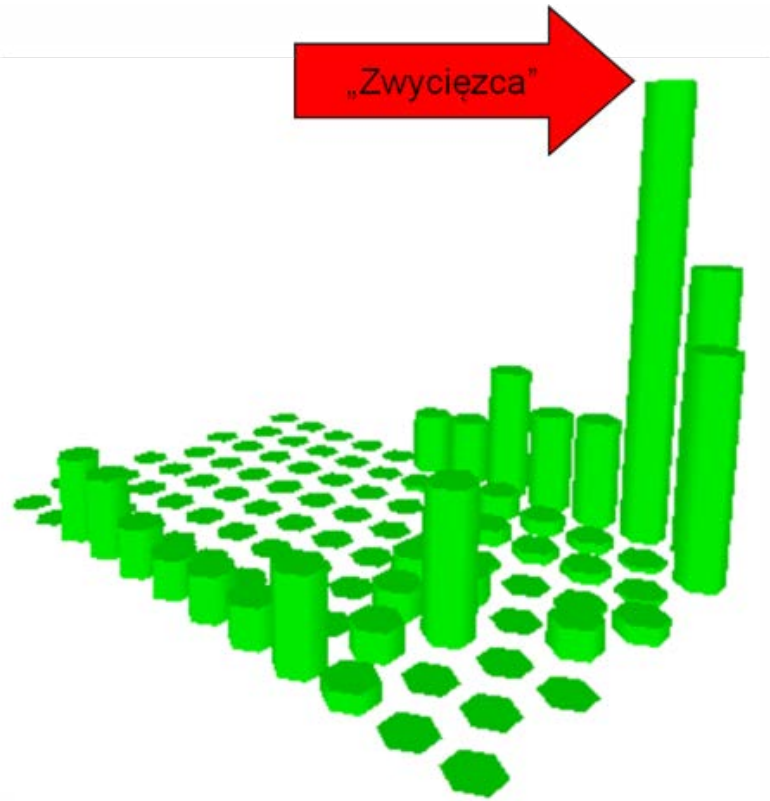
Kohonena sieć neuronowa

Najbardziej znana i najczęściej stosowana sieć **samoucząca się**, realizująca zasadę **samoorganizacji** (**SOM**). Jest to także najbardziej znany przykład sieci **konkurencyjnej** wykorzystującej koncepcję **sąsiedztwa**. W wyniku uczenia tej sieci powstaje **mapa topologiczna**, której aprioryczna interpretacja jest niemożliwa (bo sieć uczy się bez nauczyciela i użytkownik nie ma kontroli nad tym, co sieć robi). Jednak po uczeniu można zwykle ustalić, jakie znaczenie mają poszczególne rejony tej mapy (tworzonej przez **sygnały wyjściowe** pochodzące z **warstwy topologicznej**) na podstawie analizy konkretnych przykładów danych wejściowych. Strukturę sieci Kohonena przedstawia rysunek, przy czym rysunek ten ma charakter jedynie orientacyjnego schematu, bowiem rzeczywiste sieci Kohonena cechują się tym, że działają w wielowymiarowych przestrzeniach **danych wejściowych**, w związku z czym warstwa wejściowa zawiera bardzo wiele neuronów (skojarzonych z wieloma **sygnałami wejściowymi**). Podobnie typowa **warstwa topologiczna** sieci Kohonena zawiera bardzo wiele neuronów, dzięki czemu sieć po nauczeniu może prezentować bardzo subtelne rozróżnienia i klasyfikacje **danych wejściowych**.



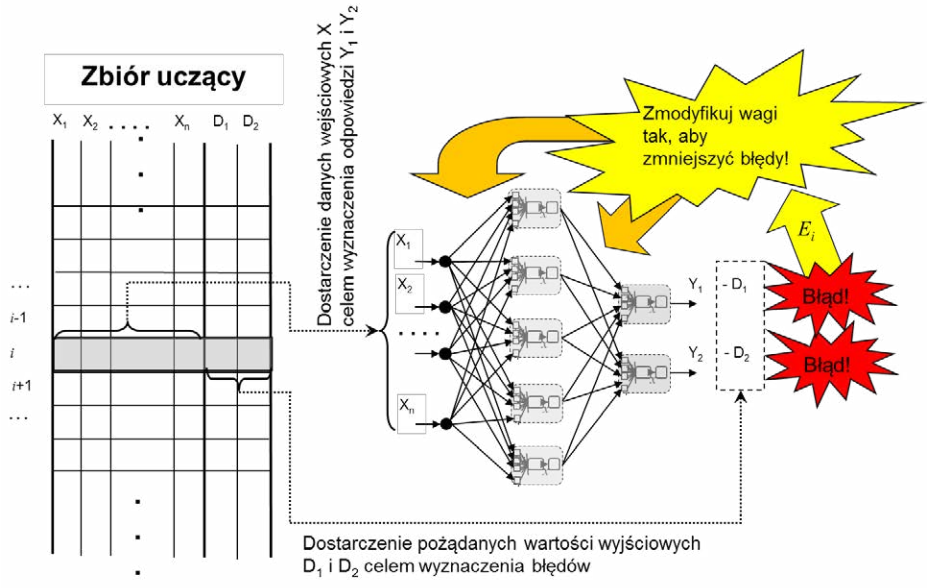
Konkurencyjna sieć neuronowa

W niektórych **sieciach neuronowych** wśród **neuronów warstwy wyjściowej** lub **mapy topologicznej** (to w przypadku sieci Kohonena) wprowadza się mechanizm konkurencji, polegający na tym, że sygnały wyjściowe tych neuronów porównuje się ze sobą. Po podaniu określonego **sygnału wejściowego** do **sieci** – na jej **wyjściu** otrzymuje się sygnały o różnych wartościach pochodzące od różnych neuronów **warstwy wyjściowej** lub **warstwy topologicznej**. Wśród tych sygnałów odnajduje się ten, który ma największą wartość i ten **neuron** zostaje wskazany jako zwycięzca (patrz rysunek). Z faktu, że określony neuron został uznany za zwycięzcę, wynikają różne konsekwencje. W szczególności w niektórych sieciach na etapie **uczenia** zmiany **wag** dotyczą wyłącznie zwycięzcy oraz (niekiedy) jego **sąsiedztwa**. W **sieciach klasyfikacyjnych** zwycięski neuron wskazuje poprawną kategoryzację **sygnału wejściowego** lub poprawne rozpoznanie obiektu reprezentowanego przez ten sygnał.



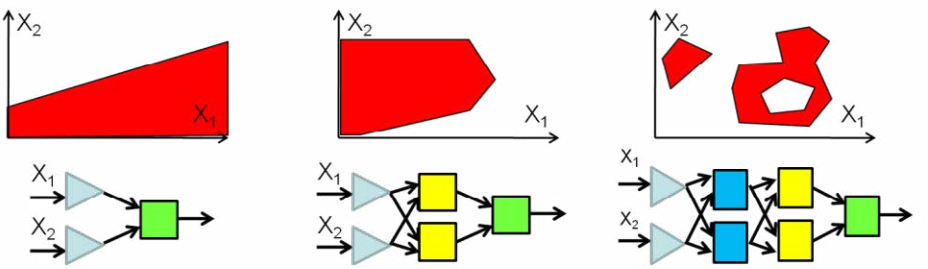
Korekta błędu

Zmiana wartości parametrów **sieci** (najczęściej **wag**) mająca na celu zmniejszenie **błędu** popełnianego przez **sieć**. Ponieważ **błąd** wyznaczany jest podczas jednego kroku procesu uczenia, przeto korekta błędu nie może być zbyt radykalna, bo łatwo jest doprowadzić do sytuacji, w której zmiana parametrów wynikająca z pokazania jednego **przypadku uczącego** ze **zbioru uczącego** może popsuć wartości parametrów ustalone wcześniej dla innych **przypadków uczących**. W praktyce wielkość korekty błędu determinuje **współczynnik uczenia**. Przebieg typowej korekty błędu przedstawia poniższy schemat.



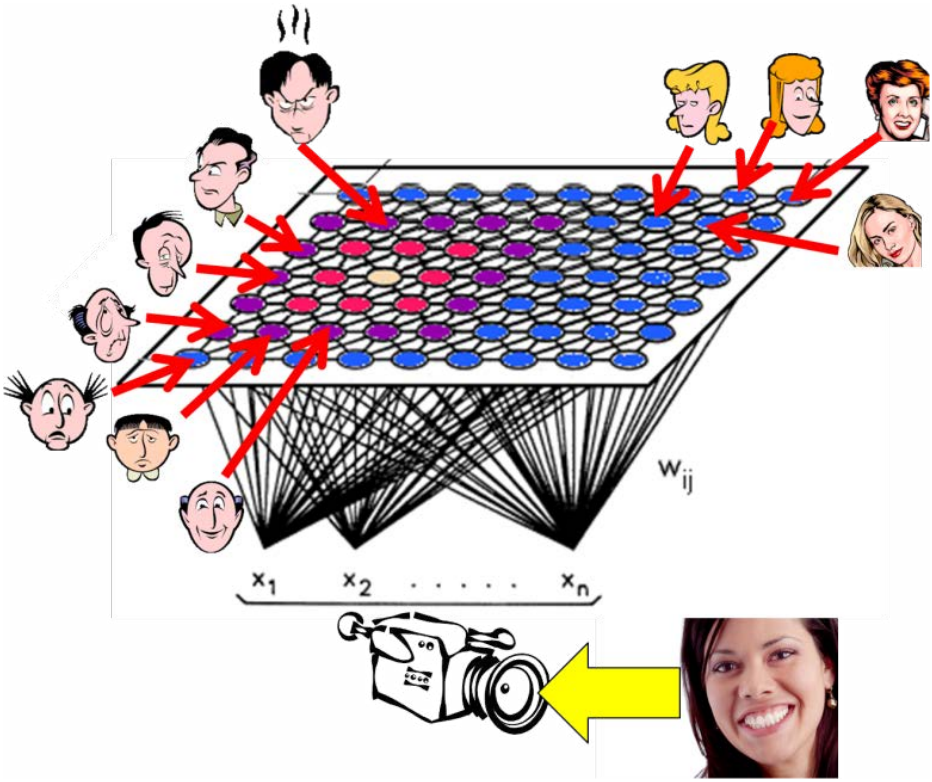
Liczba warstw ukrytych

W **strukturze sieci neuronowej** może występować różna liczba **warstw ukrytych**. Decyzję o tym, ile **warstw ukrytych** zastosować, podejmuje twórca sieci i jest to na ogół decyzja arbitralna. Tym bardziej trzeba sobie zdawać sprawę z konsekwencji tej decyzji. Na rysunku przedstawiono w sposób umowny kształty **obszarów decyzyjnych**, jakie mogą tworzyć sieci o różnej liczbie warstw ukrytych. Widać, że sieci bez warstwy ukrytej mogą dzielić przestrzeń **sygnałów wejściowych** na dwie części, rozgraniczone linią prostą (w przypadku wielowymiarowym, to znaczy gdy sieć ma wiele wejść – *hiperpłaszczyzną*). Sieć mająca jedną **warstwę ukrytą** może wydzielić w przestrzeni sygnałów wejściowych dowolny jednospójny obszar o wypukłym obrysie, nazywany *simpleksem*. Dopiero sieć mająca dwie warstwy ukryte pozwala budować **obszary decyzyjne** otoczone niewypukłą powierzchnią graniczną, a także niejednospójne. Zastosowanie jeszcze większej liczby warstw ukrytych już bardziej **obszarów decyzyjnych** wzbogacić nie może, więc jest niecelowe.



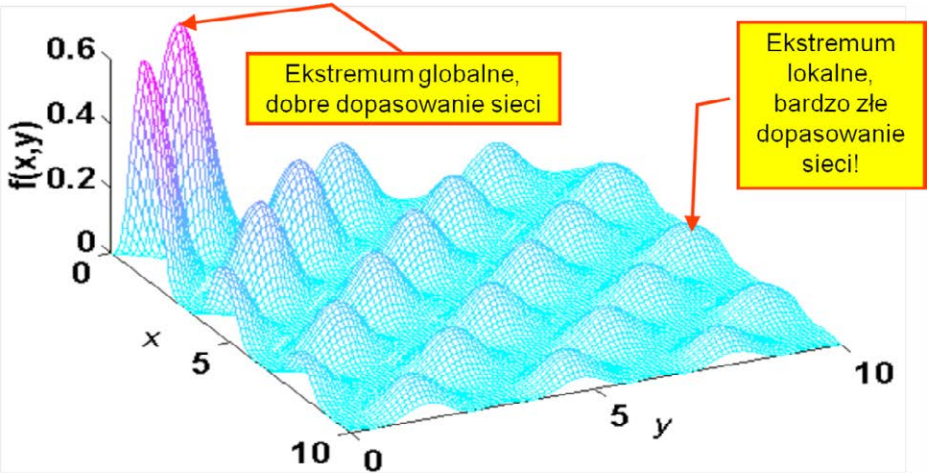
Mapa topologiczna

W **sieci Kohonena** ta **warstwa**, na której prezentowany jest wynik działania **sieci**, nazywana jest **warstwą topologiczną**. **Neurony** należące do tej warstwy specjalizują się w identyfikowaniu poszczególnych obiektów, jakie w trakcie procesu **samouczenia** były sieci prezentowane na jej **wejściu**. Każdy neuron **warstwy topologicznej** ma więc przypisany do siebie obiekt, którego pojawienie się na wejściu sieci powoduje, że ten właśnie neuron zostaje zwycięzcą (patrz hasło **Konkurencyjna sieć neuronowa**). Rozmieszczenie tych obiektów formuje właśnie mapę topologiczną, pokazaną symbolicznie na rysunku. Znajomość mapy topologicznej ułatwia użytkownikowi interpretację i wykorzystanie wyników obliczeń dostarczanych przez sieć Kohonena.



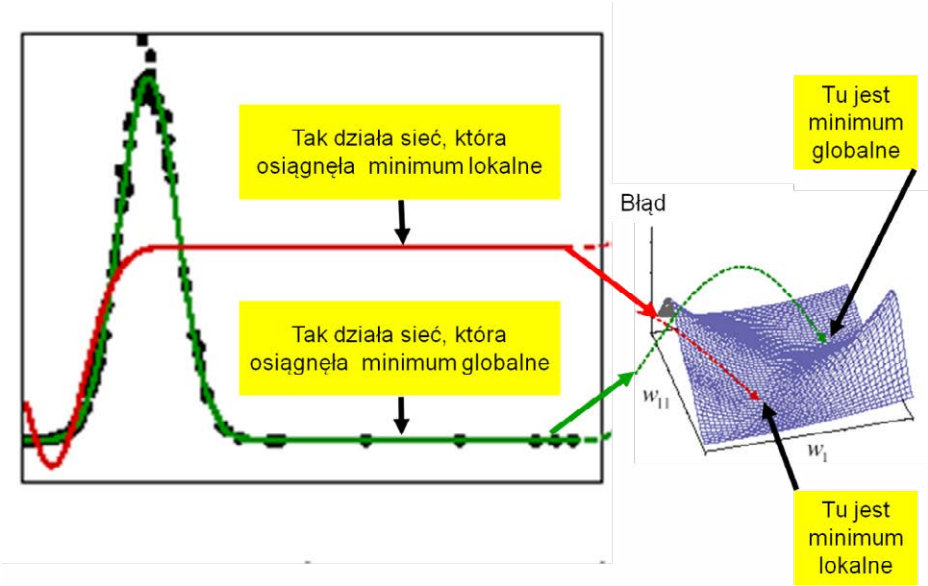
Minimum globalne

Efekt **uczenia** sieci neuronowej tylko wtedy jest dobry, jeśli w jego wyniku wszystkie **wagi** otrzymają wartości gwarantujące uzyskanie najmniejszej wartości **błędu** całej **sieci**. Na wykresie **funkcji błędu** odpowiada to znalezieniu minimum globalnego. Na rysunku **funkcję błędu** odwrócono, bo łatwiej jest narysować maksimum niż minimum, ale oczywiście w **uczeniu sieci neuronowych** zmierzamy do znalezienia minimum.



Minimum lokalne

Proces uczenia sieci neuronowej jest w istocie procesem minimalizacji **funkcji błędu**. Przy każdej minimalizacji istnieje niebezpieczeństwo, że w wyniku otrzymane zostanie minimum lokalne, a nie **minimum globalne**. Na rysunku pokazano, czym to grozi: otóż **sieć**, która wpadła w „pułapkę" minimum lokalnego, źle aproksymuje **dane wyjściowe** (zaznaczone na rysunku czarnymi prostokątami).

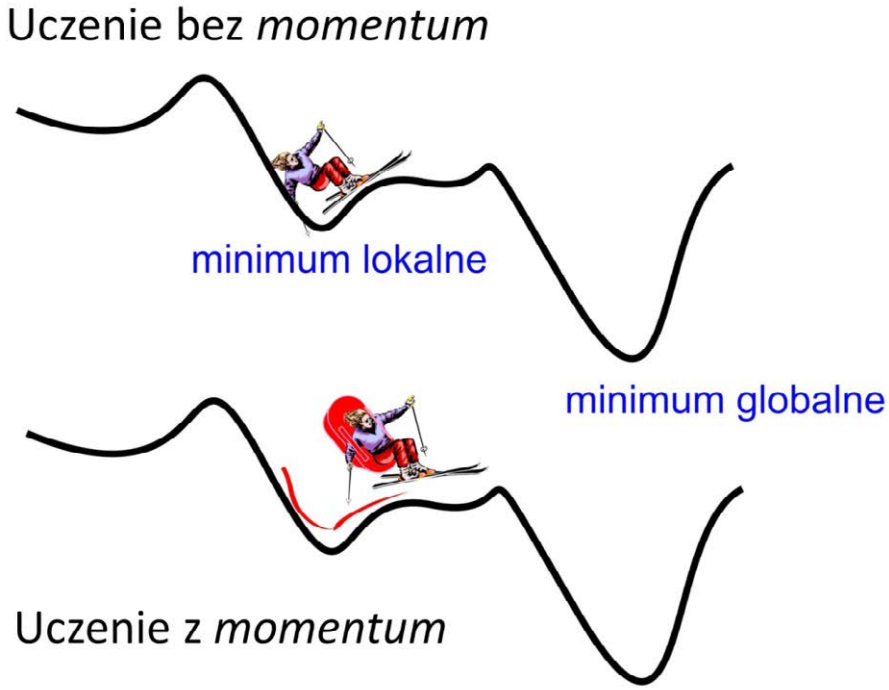


MLP

Patrz hasło: **Perceptron wielowarstwowy**.

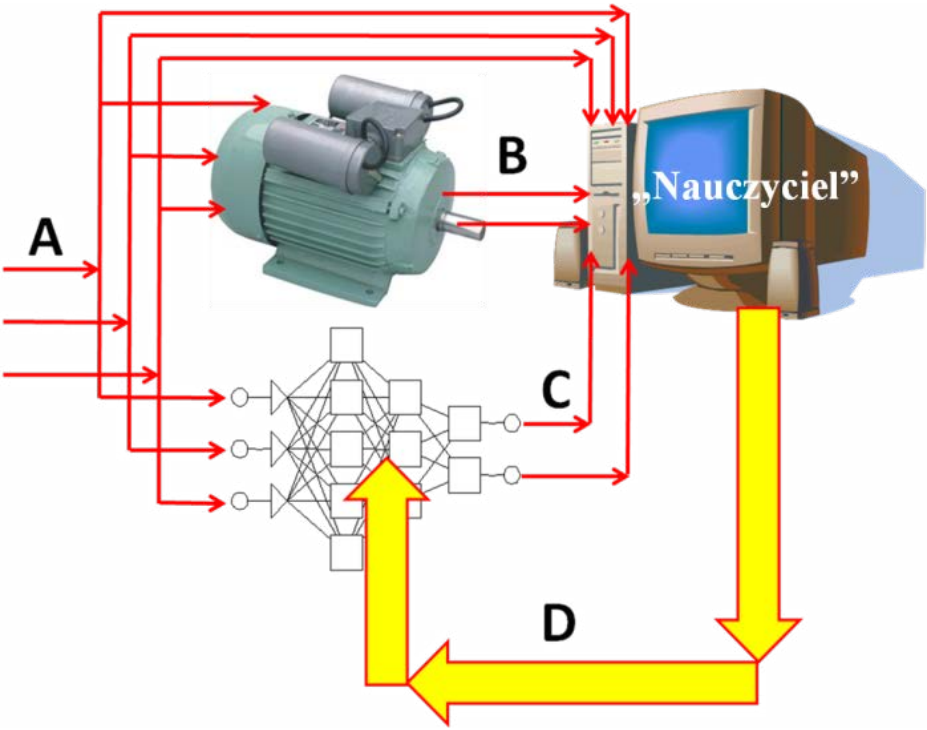
Momentum

Składnik algorytmu uczenia powodujący, że podczas procesu **uczenia** zmiany wartości **wag** nie następują natychmiast po zmianie gradientu błędu, ale z pewną bezwładnością. Kierunek zmian **wag** jest więc dłużej zachowywany (uczenie jest bardziej konsekwentne), a proces uczenia jest mniej wrażliwy na pułapki **minimów lokalnych**.



Nauczyciel

Podmiot realizujący **algorytm uczenia**. Pojęcie 'nauczyciela' w przypadku **uczenia sieci neuronowej** jest pewną metaforą. W istocie nie ma żadnego człowieka, który mógłby pełnić rolę nauczyciela w stosunku do sieci neuronowej, ponieważ dla wytworzenia w sieci wymaganej wiedzy potrzeba setek, a czasem tysięcy pokazów **przykładów uczących** (na rysunku są to różne stany pracy modelowanej przez sieć maszyny) i korekt **wag** (na rysunku obrazują to żółte strzałki). Dlatego „nauczycielem" jest zawsze komputer mający możność wykonywania wszystkich czynności związanych z uczeniem sieci w sposób automatyczny i z nieograniczoną cierpliwością. Ale przy opisywaniu tego, jak sieć się uczy i jak działa – wygodnie jest powoływać się na metaforę *nauczyciela*, więc jest ona powszechnie stosowana.



Oznaczenia na rysunku:

A – sygnały wejściowe kierowane do obiektu, którego model ma wytworzyć w toku uczenia sieć neuronowa. Sygnały te są również przekazywane do modelującej obiekt sieci neuronowej oraz do komputera, który pełni rolę „nauczyciela".

B – sygnały wyjściowe produkowane przez modelowany obiekt. Są one rejestrowane w komputerze pełniącym rolę nauczyciela jako wzorzec poprawnej odpowiedzi, jaką powinna wyprodukować sieć.

C – sygnały produkowane przez sieć, także rejestrowane w komputerze pełniącym rolę nauczyciela.

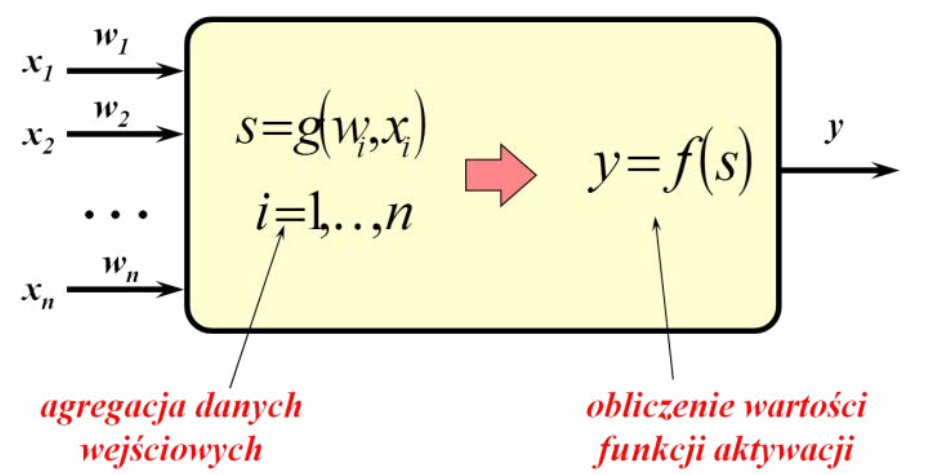
D – oddziaływanie korekcyjne zmieniające parametry sieci neuronowej, ustalane w komputerze pełniącym rolę nauczyciela na podstawie algorytmu uczenia i porównania sygnałów B i C, które powinny stawać się coraz bardziej podobne do siebie. Przy realizacji algorytmu uczenia uwzględniane są także sygnały wejściowe A.

Neuron

Podstawowy element budujący **strukturę sieci neuronowej**. Jest to element przetwarzający informacje, w pewnym stopniu wzorowany na funkcjonowaniu biologicznej komórki nerwowej, ale bardzo uproszczony.

Z powodu tych uproszczeń w zasadzie nie powinno się używać dla tych elementów nazwy 'neuron', bo ich właściwości daleko odbiegają od prawdziwych komórek nerwowych i ich dokładnych modeli (na przykład dostępnych w programie GENESIS). Ale nazwa neuron przyjęła się i jest powszechnie używana.

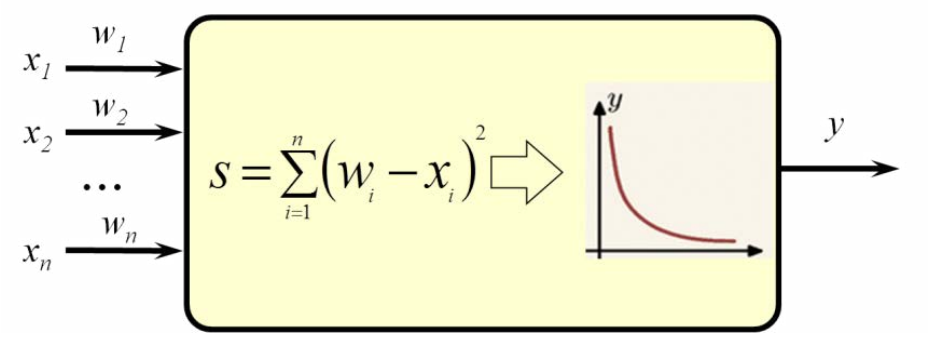
W strukturze **neuronu** odnaleźć można wiele **wejść** oraz jedno **wyjście**. Ważnym składnikiem neuronu jest komplet **wag**, których wartości decydujące o zachowaniu neuronu zazwyczaj ustalane są w trakcie procesu **uczenia**.



W neuronie wykonywane są zwykle dwie czynności: **agregacja danych wejściowych** (z uwzględnieniem **wag**) oraz generacja sygnału wyjściowego (**danej wyjściowej**). Ze względu na sposób **agregacji** oraz formę **funkcji aktywacji** wyróżnia się różne typy **neuronów**. Najczęściej stosowane są **neurony liniowe**, **neurony sigmoidalne** i **neurony radialne**. Odmianą neuronów sigmoidalnych są **neurony tangensoidalne**.

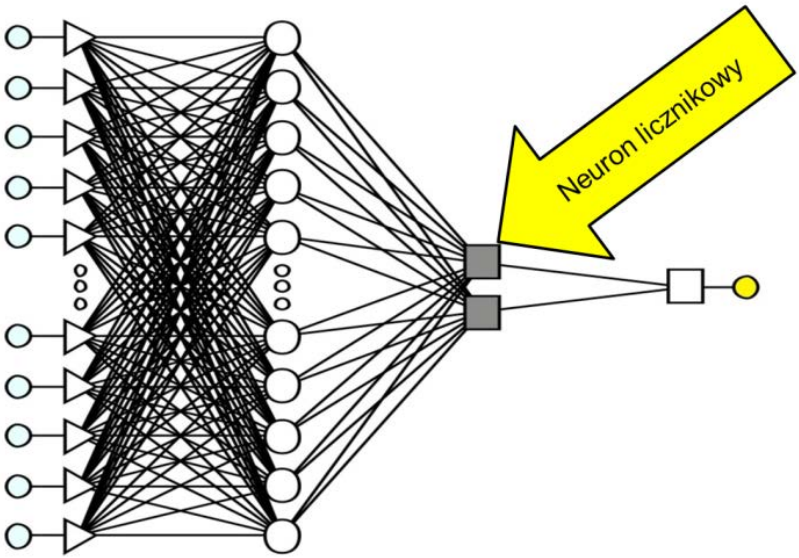
Neuron Kohonena

W **sieciach Kohonena** używane są neurony bardzo podobne pod względem struktury i funkcji do **neuronów radialnych** – ale różniące się jednym szczegółem, który może mieć znaczenie. Neurony te dokonują **agregacji sygnałów wejściowych** zgodnie ze schematem agregacji radialnej, ale jako **funkcja aktywacji** używana jest *odwrotność* sygnału **sumarycznego pobudzenia**. Neurony Kohonena reagują więc bardzo dużym sygnałem wyjściowym w sytuacji, gdy występuje mała odległość wektora **danych wejściowych** i wektora **wag**. Przy rosnącej odległości tych dwóch wektorów **sygnał wyjściowy neuronu** szybko maleje i utrzymuje niewielką wartość dla wszystkich **wektorów wejściowych** z wyjątkiem tych właśnie, które są bardzo bliskie **wektora wag**. Przy praktycznej realizacji neuronów Kohonena trzeba wprowadzić do **funkcji aktywacji** dodatkowe zabezpieczenie na wypadek, gdyby **wektor wejściowy** pokrył się z **wektorem wag**. W związku z tym w mianowniku wyrażenia definiującego **funkcję aktywacji** musi być dodana mała stała wartość (na przykład 10^{-8}), która normalnie nie ma znaczenia, bo jest znacznie mniejsza od znajdującej się także w tym mianowniku odległości **wektora wejść** i **wektora wag** (wyznaczonej przez agregację radialną). Stała ta zabezpiecza jednak przed fatalnymi skutkami dzielenia przez zero, gdy wektory **wejść** i **wag** przypadkowo się pokryją. Schemat neuronu Kohonena pokazano na rysunku.



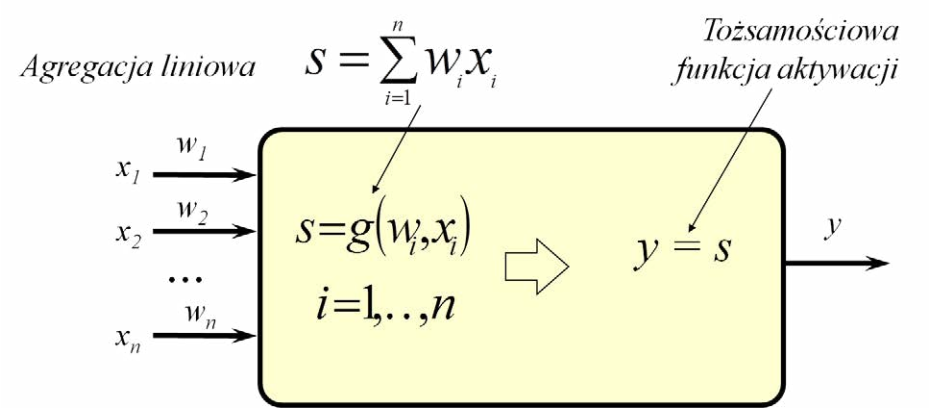
Neuron licznikowy

Jest to neuron wykorzystywany w drugiej **warstwie ukrytej** sieci **uogólnionej regresji**. Neuron licznikowy oblicza iloczyn skalarny własnego wektora **wag** i wektora sygnałów pochodzących z warstwy radialnej. **Wagi** tego neuronu są tak ustalane, że każdy sygnał warstwy radialnej zostaje przemnożony przez sumę wektorów wejściowych (przypadków) rozpoznawanych przez dany **neuron radialny**.



Neuron liniowy

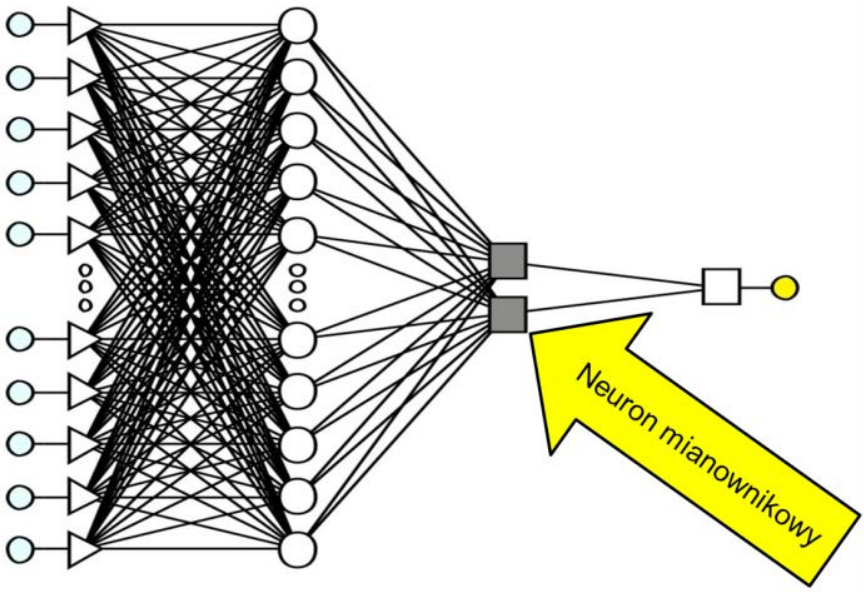
Jest to najprostszy, ale często bardzo przydatny model neuronu. Zakłada on liniową **agregację danych wejściowych** i tożsamościową (liniową) **funkcję aktywacji**.



Sieci zbudowane z neuronów liniowych bardzo dobrze się uczą, ale ich możliwości są ograniczone i nie każde zadanie da się rozwiązać przy ich pomocy.

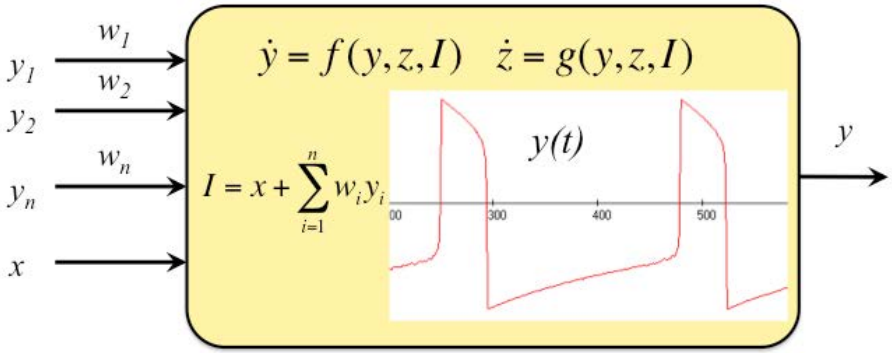
Neuron mianownikowy

Jest to neuron wykorzystywany w drugiej **warstwie ukrytej** sieci **uogólnionej regresji**. Neuron mianownikowy oblicza iloczyn własnego wektora **wag** i wektora **sygnałów wejściowych**, przy czym wagi tego neuronu są ustalane w procesie uczenia w taki sposób, że są proporcjonalne do ilości przypadków rozpoznawanych przez dany wektor radialny.



Neuron oscylacyjny (impulsujący)

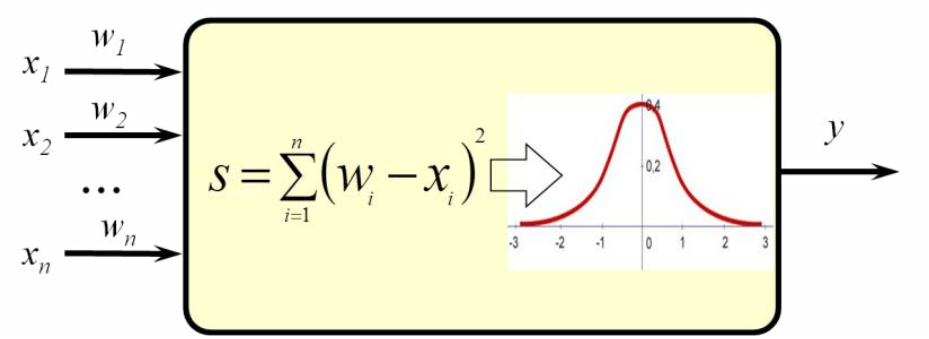
Neurony impulsujące są bardziej złożoną odmianą neuronów „klasycznych". Zwykle ich działanie opisuje układ równań różniczkowych zwyczajnych, które stanowią model matematyczny oscylatora relaksacyjnego. Oznacza to, że **wyjście** takiego **neuronu** generuje **sygnał** okresowy, który przy odpowiednim **sumarycznym pobudzeniu** jest zbliżony do oscylacji występujących w tkance nerwowej ludzkiego mózgu. Dlatego różne rodzaje neuronów impulsujących posłużyły do budowania sieci modelujących, naturalnie w sposób przybliżony, zachowanie ludzkiej kory mózgowej. **Sieci** takie są wykorzystywane np. do analizy sceny wizyjnej, a w szczególności do segmentacji obrazów.



Pobodzenie neuronu oscylacyjnego, poza sygnałem wejściowym, stanowi również suma ważonych **wyjść** sąsiednich **neuronów**. Dzięki tym lokalnym sprzężeniom neurony mają możliwość wzajemnej synchronizacji, co jest ich ważną cechą. W przypadku sieci zbudowanej z takich neuronów, grupy neuronów jednocześnie pobudzonych (zsynchronizowanych, czyli oscylujących w tym samym czasie) mogą kodować np. jednorodne obszary analizowanego obrazu.

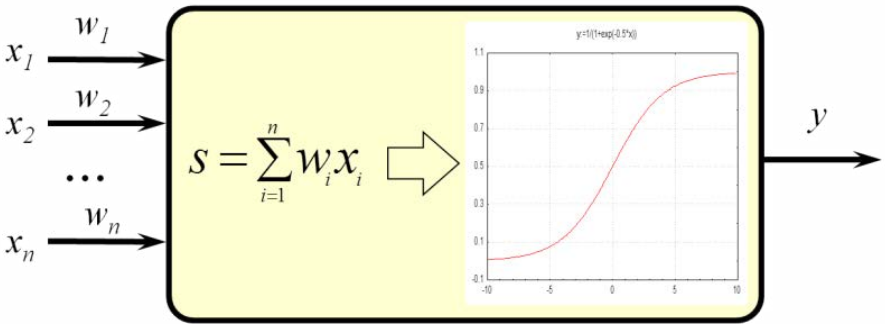
Neuron radialny

Neurony radialne stosowane są w **sieciach radialnych RBF** oraz w sieciach **uogólnionej regresji** określanych też jako **GRNN**. Struktura neuronu radialnego zakłada użycie radialnej **agregacji danych wejściowych** oraz funkcji Gaussa jako **funkcji aktywacji**.



Neuron sigmoidalny

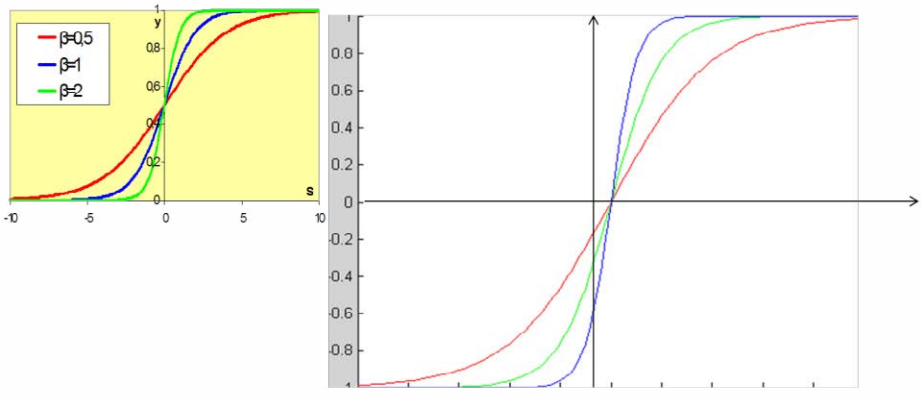
Jest to najbardziej popularny **neuron** nieliniowy, nadający się do budowy sieci **MLP**. W neuronie sigmoidalnym zastosowana jest liniowa **agregacja danych wejściowych** (często z uwzględnieniem składnika **BIAS**) oraz sigmoidalna **funkcja aktywacji**. Na marginesie można dodać, że schemat działania neuronu sigmoidalnego jest najbardziej zbliżony do działania prawdziwej biologicznej komórki nerwowej.



Warto jeszcze raz podkreślić, że zdecydowana większość dobrze funkcjonujących sieci neuronowych, wykorzystywanych w praktyce w różnych dziedzinach. wykorzystuje w swojej **strukturze**, a zwłaszcza w **warstwach ukrytych**, składniki w postaci neuronów sigmoidalnych.

Neuron tangensoidalny

Neuron sigmoidalny, najbardziej popularny i najczęściej stosowany, ma ograniczenie polegające na tym, że jego **sygnał wyjściowy** może przyjmować wyłącznie wartości dodatnie. Taką funkcję aktywacji nazywa się często *unipolarną*. Biologiczne komórki nerwowe, na których wzorowane są wszystkie **neurony** wykorzystywane do budowy **sieci neuronowych**, mogą operować wyłącznie dodatnimi sygnałami, więc wybór sigmoidalnej **funkcji aktywacji** jest wyborem zmierzającym do zapewnienia biologicznej wierności **sieci**. Jednak w sztucznej **sieci neuronowej** możliwe jest używanie zarówno sygnałów unipolarnych (wyłączenie dodatnich) jak i bipolarnych (zarówno dodatnich, jak i ujemnych). Z tego powodu w sieciach neuronowych zaczęto stosować neurony posiadające funkcję aktywacji opisaną przez formułę tangensa hiperbolicznego. Jak pokazano na rysunku, funkcja ta ma taki sam kształt jak sigmoida, jednak jej wartości rozciągają się od -1 do +1, podczas gdy wartości sigmoidy są rozpięte między 0 a +1.



Sigmoida

$$f(s) = \frac{1}{1+\exp(-\beta s)}$$

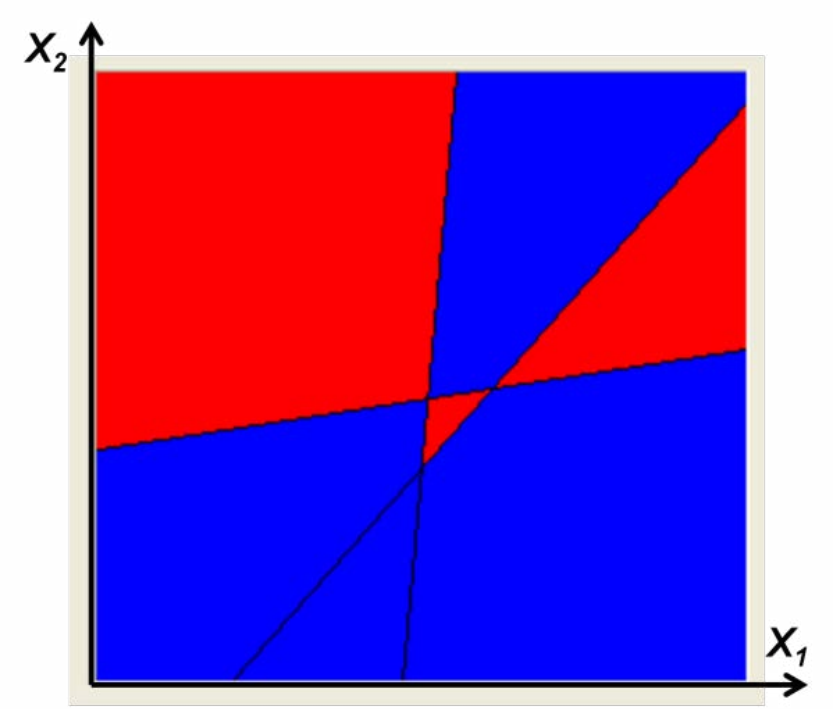
Funkcja tangens hiperboliczny

$$f(s) = \tanh(\beta s) = \frac{\exp(\beta s) - \exp(-\beta s)}{\exp(\beta s) + \exp(-\beta s)}$$

Niestety nie spełniło się oczekiwanie, że bipolarna funkcja tangens hiperboliczny użyta jako **funkcja aktywacji** w miejsce sigmoidy przyniesie znaczące polepszenie działania sieci neuronowych. Dlatego neurony tangensoidalne są rzadziej używane niż **sigmoidalne**.

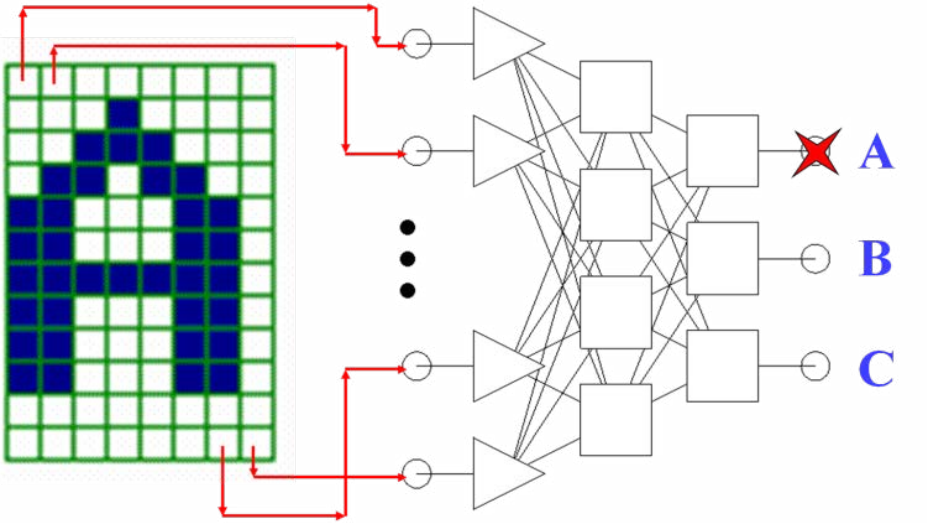
Obszary decyzyjne

Przy dyskutowaniu właściwości sieci neuronowych użyteczne bywa przedstawienie w układzie współrzędnych wyznaczanych przez **dane wejściowe** obszarów, w których **sygnał wyjściowy** sieci przyjmuje przeciwstawne wartości – na przykład +1 i -1. Obszary te, określane jako obszary decyzyjne, w sposób dokładny opisują zachowanie jedynie takiej **sieci**, która ma dwa **wejścia** i jedno **wyjście**. Jednak szkice obszarów decyzyjnych bywają przydatne także przy *jakościowej* ocenie zachowania sieci neuronowych o znacznie bogatszej strukturze. Na rysunku pokazano przykładowe obszary decyzyjne prostej sieci o dwóch wejściach i jednym wyjściu. Do obszarów, które powodują pojawienie się sygnału +1 na wyjściu sieci, przypisano kolor czerwony, a do obszarów, które powodują pojawienie się sygnału -1 na wyjściu sieci, przypisano kolor niebieski.



Odpowiedź

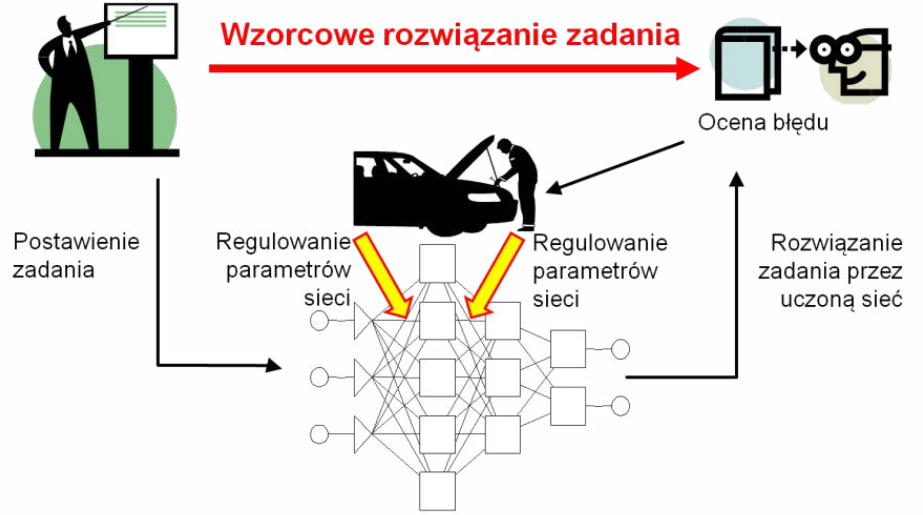
Skrótowe określenie używane jako synonim **danych wyjściowych**. Można rozważać odpowiedź pojedynczego neuronu oraz odpowiedź całej sieci, utożsamianą z kompletem **danych wyjściowych** dla wszystkich **neuronów** wchodzących w skład **warstwy wyjściowej** sieci.



Odpowiedź sieci wymaga często dodatkowej interpretacji – na przykład w **sieciach klasyfikacyjnych** rozpoznanie ustala się na podstawie tego, który **neuron** warstwy wyjściowej prezentuje największą wartość **sygnału**.

Odpowiedź wzorcowa

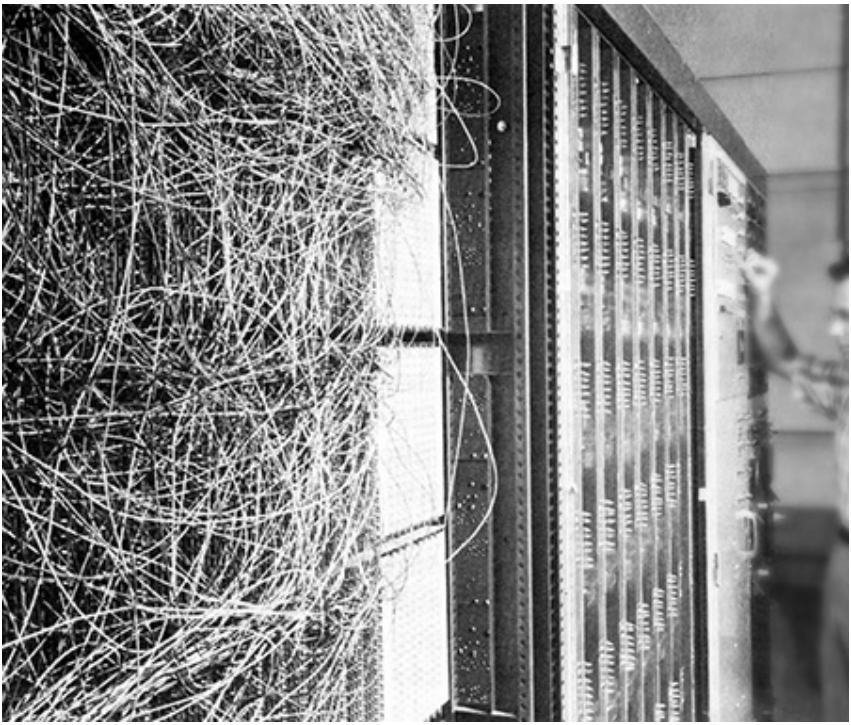
Druga składowa każdego **przypadku uczącego** należącego do **zbioru uczącego**. Jest to wzorzec poprawnych **danych wyjściowych** (poprawnej odpowiedzi sieci) dla **przykładowych danych wejściowych** stanowiących pierwszą składową tego **przypadku uczącego**. Odpowiedź wzorcowa jest wykorzystywana do wyznaczania **błędu** w czasie wykonywania **algorytmu uczenia** sieci. Ilustrację wykorzystania odpowiedzi wzorcowej przy **uczeniu sieci** przedstawiono na rysunku.



Perceptron

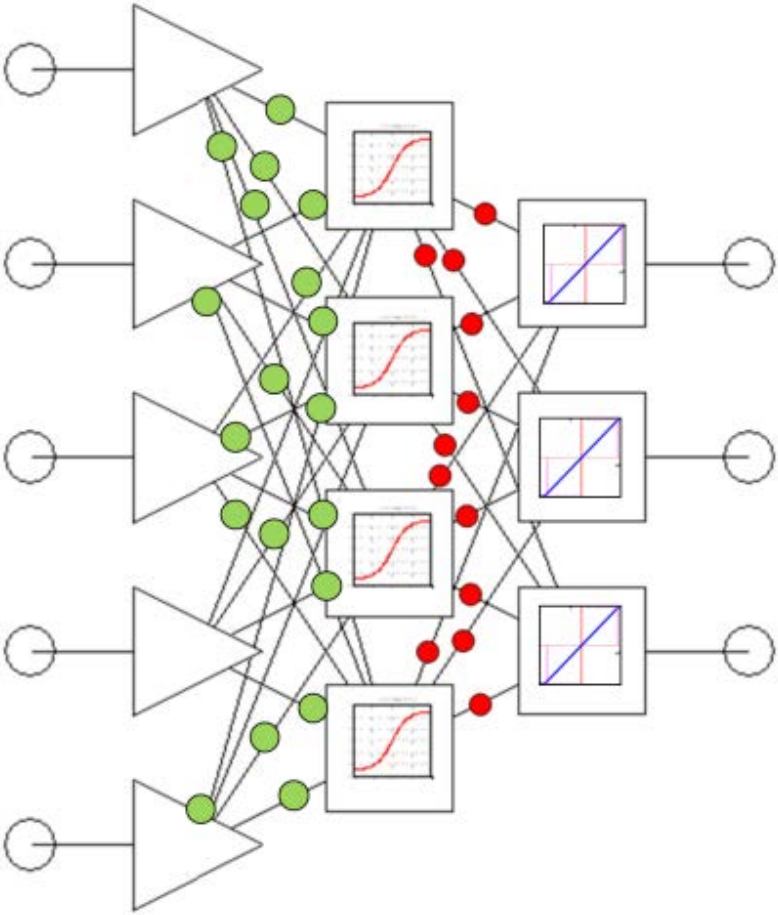
Nazwa kojarzona często z **jednokierunkowymi sieciami neuronowymi** uczonymi metodą **uczenia** z **nauczycielem**, które są przeznaczone do rozpoznawania i klasyfikacji różnych obiektów (najczęściej obrazów). Nazwa ta została po raz pierwszy użyta dla określenia sprzętowej elektromechanicznej **sieci neuronowej**, którą zbudował i przebadał w 1960 roku Frank Rosenblatt na Uniwersytecie Cornella.

Wygląd tej sieci przedstawia fotografia zaczerpnięta w dniu 15 lutego 2014 roku ze strony http://www.rutherfordjournal.org/images/TAHC_perceptron.jpg. Na pierwszym planie widać kable realizujące **połączenia** między **neuronami** (widocznymi w głębi fotografii w postaci modułów o budowie elektromechanicznej – zmianę **wag** podczas **uczenia** uzyskiwano poprzez silniki elektryczne, które obracały potencjometry). Z obrazu **połączeń** widać, że w Perceptronie neurony miały *przypadkowe* **połączenia**, a jednak ta **sieć** po **procesie uczenia** realizowała poprawnie stawiane jej zadania (rozpoznawanie znaków pisma, figur geometrycznych itp.).



Perceptron wielowarstwowy

Jest to bardzo popularny typ sieci jednokierunkowej, kojarzony również ze skrótem **MLP** (od *Multilayer Perceptron*). Sieć typu MLP ma zwykle **strukturę** obejmującą warstwy: **wejściową**, jedną lub dwie **warstwy ukryte** złożone z **neuronów sigmoidalnych** oraz **warstwę wyjściową** złożoną z **neuronów sigmoidalnych** lub z **neuronów liniowych**. **Uczenie** perceptronu wielowarstwowego realizowane jest najczęściej przy użyciu metody **wstecznej propagacji błędów**. Na rysunku wewnątrz kwadratów reprezentujących neurony narysowano wykresy przywołujące odpowiednie **funkcje aktywacji**, a kółkami oznaczono podlegające **procesowi uczenia wagi**.



PNN

Patrz hasło: **Probabilistyczna sieć neuronowa**.

Podział przypadków uczących na podgrupy

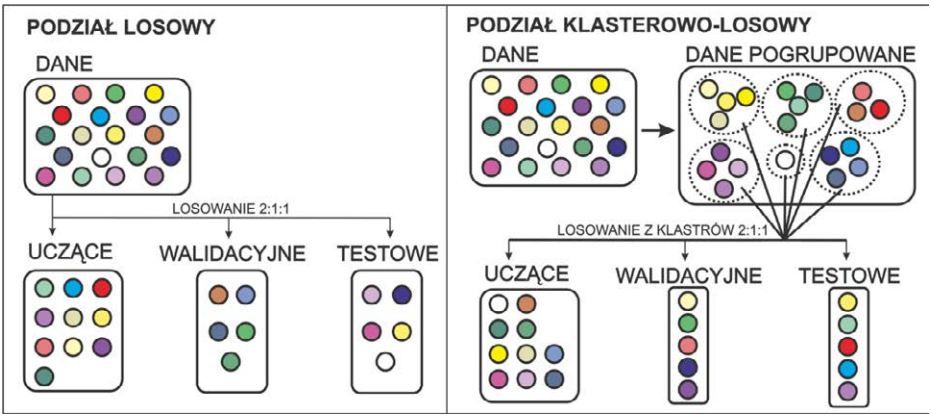
Aby przeprowadzić **uczenie sieci neuronowej** z **nauczycielem** należy podzielić dane przynajmniej na dwie części: **zbiór uczący** i **walidacyjny**. Bardzo często wydziela się ze zbioru również trzeci podzbiór przypadków, **zbiór testowy**, służący do ostatecznej oceny jakości sieci (zdolności **generalizacyjnych**).

Główny problemem związany z wydzieleniem podzbiorów polega na konieczności posiadania w każdym zbiorze **przypadków uczących** reprezentatywnych dla całego zbioru. Jeżeli ze zbioru uczącego zostaną wydzielone przypadki unikatowe, model nie będzie w stanie poprawnie przewidzieć ich własności. Z drugiej strony, jeżeli zostaną wybrane przypadki standardowe, posiadające bardzo bliskie lub niemal identyczne odpowiedniki w zbiorze uczącym, cała procedura oceny jakości sieci będzie nieskuteczna, gdyż nawet model **przeuczony** uzyska bardzo dobre predykcje w czasie walidacji i testowania.

Można wyróżnić 2 sposoby podziału przypadków na podzbiory:

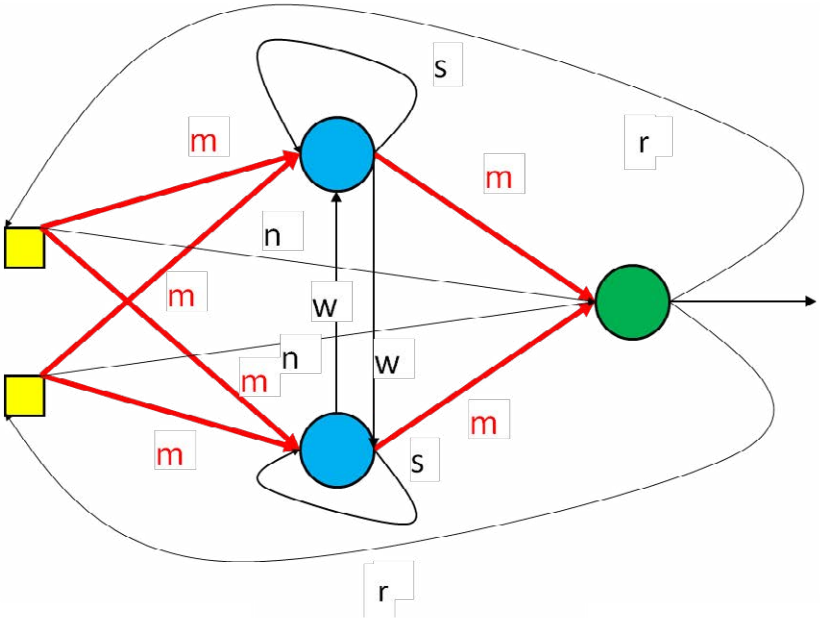
- Losowy (np. w stosunku 2:1:1) – poprawny w przypadku bardzo licznego zbiór przypadków
- Klasterowy – bazujący na podobieństwach w samej strukturze danych.

Należy dobrać empirycznie liczbę klastrów do zbioru danych, umożliwiając proporcjonalny wybór reprezentantów z poszczególnych klastrów w każdym z podzbiorów. W przypadku, gdy niezależnie od liczby klastrów zawsze obserwujemy występowanie pojedynczych unikatowych przypadków (tj. klastrów zawierających jeden przypadek), należy je włączyć do zbioru uczącego. Wyboru reprezentantów można dokonywać losowo albo kierując się odległością od centrum klastra


PODZIAŁ LOSOWY
DANE
LOSOWANIE 2:1:1
UCZĄCE
WALIDACYJNE
TESTOWE
PODZIAŁ KLASTEROWO-LOSOWY
DANE
DANE POGRUPOWANE
LOSOWANIE Z KLASTRÓW 2:1:1
UCZĄCE
WALIDACYJNE
TESTOWE

Połączenia

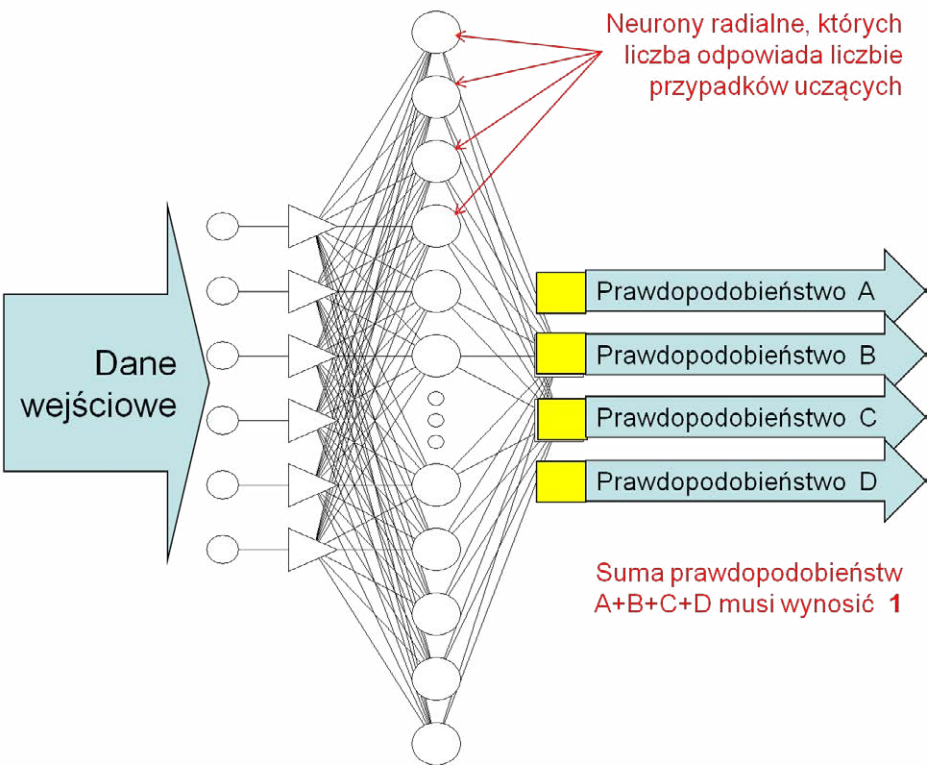
Sieć neuronowa powstaje w ten sposób, że pomiędzy **wyjściami** jednych **neuronów** a **wejściami** innych **neuronów** tworzone są połączenia służące do jednokierunkowego przesyłania sygnałów (danych). Są możliwe (patrz rysunek): (m) – połączenia międzywarstwowe, (w) – połączenia wewnątrzwarstwowe, (n) – połączenia nadwarstwowe, (s) – samosprzężenia, (r) – połączenia rekurencyjne. W większości praktycznie używanych sieci występują tylko połączenia międzywarstwowe, przy czym stosowana jest zasada **każdy z każdym**.



Zazwyczaj z każdym połączeniem związana jest **waga**, której wartość uczestniczy w procesie **agregacji danych** w **neuronie**. Wartość **wagi** może ulegać zmianie w trakcie procesu uczenia.

Probabilistyczna sieć neuronowa

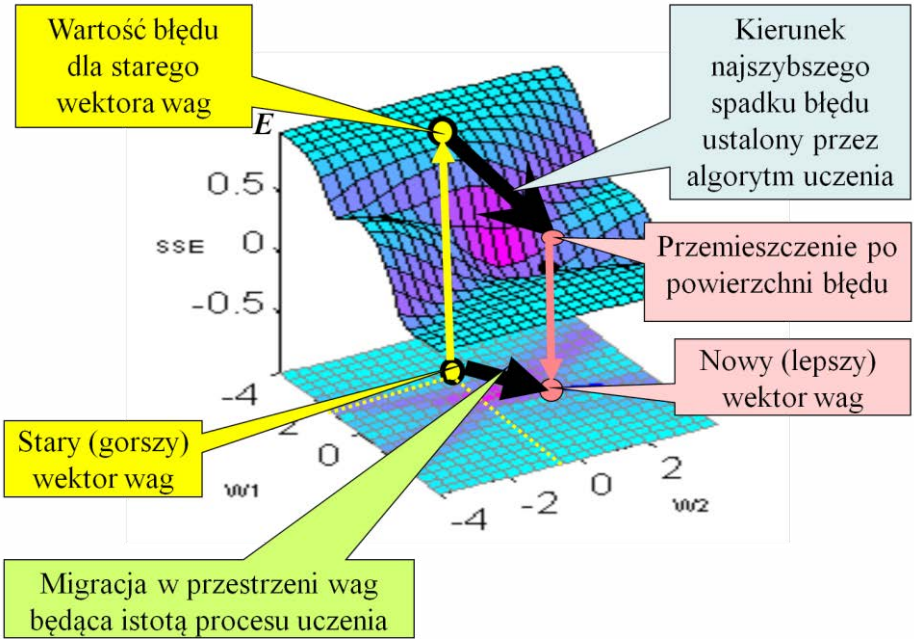
Są to **sieci neuronowe**, w których **wyjścia** traktowane są jako prawdopodobieństwa poszczególnych możliwych rozwiązań. Sieci tego typu określane są często skrótem **PNN** (*Probabilistic Neural Networks*). Są to **sieci radialne** zwykle o liczbie neuronów w **warstwie ukrytej** równej liczbie **przypadków uczących**. Zasadniczą cechą sieci probabilistycznych jest takie normalizowanie wartości **sygnałów wyjściowych**, że ich suma (na wszystkich **wyjściach** sieci) ma wartość 1. Wówczas można przyjąć, że wartości na poszczególnych **wyjściach** sieci reprezentują prawdopodobieństwa kategorii (rozpoznań) przypisanych do tych wyjść.



Proces uczenia

Istota procesu uczenia polega na tym, że **nauczyciel**, realizując **algorytm uczenia**, modyfikuje **wagi** we wszystkich **neuronach** sieci neuronowej w taki sposób, żeby zmierzać do minimum **funkcji błędu**. **Algorytm uczenia** określa tylko sposób *polepszania* zestawu **wag**. Działa on w ten sposób, że, mając przed wykonaniem kolejnego kroku procesu uczenia gorszy zestaw **wag**, dokonuje takiej jego zmiany, żeby uzyskać lepszy (to znaczy gwarantujący mniejszy błąd) nowy zestaw **wag**. Ten proces wymaga **inicjalizacji wag** (najczęściej losowej – patrz hasło **inicjalizacja**).

Przy realizacji procesu uczenia ważne jest, żeby udało się znaleźć **minimum globalne**, gwarantujące rzeczywiście najlepsze dopasowanie parametrów sieci do rozwiązywanego zadania, z równoczesnym unikaniem pułapek, jakimi są liczne na ogół **minima lokalne**.



O dynamice procesu uczenia decyduje **współczynnik uczenia** oraz ewentualnie bezwładność uczenia, wiązana zwykle z angielskim terminem **momentum**.

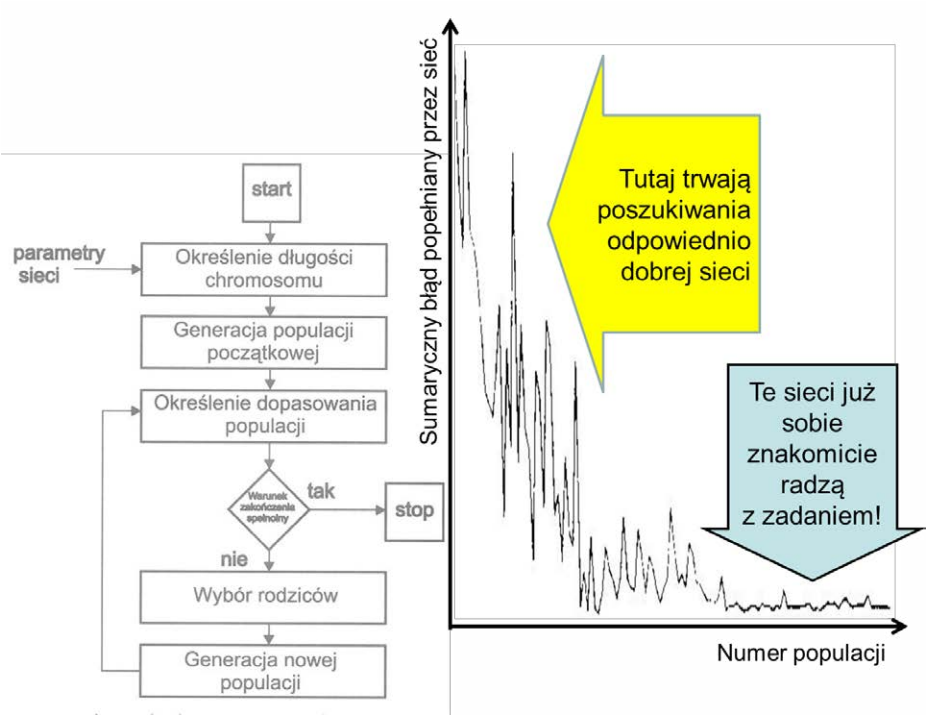
Programy modelujące sieci neuronowe

Koncepcja **sieci neuronowej** jako narzędzia przetwarzającego informacje może być realizowana na dwa sposoby. Możliwa jest budowa specjalistycznych urządzeń, w których sieć neuronowa jest specjalizowanym układem elektronicznym, ale najczęściej sieć neuronowa jest po prostu programem komputerowym umożliwiającym definiowanie **struktury sieci**, przeprowadzenie procesu **uczenia** sieci oraz jej eksploatację jako narzędzia informatycznego do rozwiązywania określonych problemów.

Do bardziej znanych programów modelujących sieci neuronowe należą *Neural Network Toolbox* do MATLABa oraz *Statistica Neural Networks* w programie *Statistica*. Darmowe programy do modelowania sieci neuronowych (wraz z kodami źródłowymi w języku C#) można pobrać ze strony http://home.agh.edu.pl/~tad//.

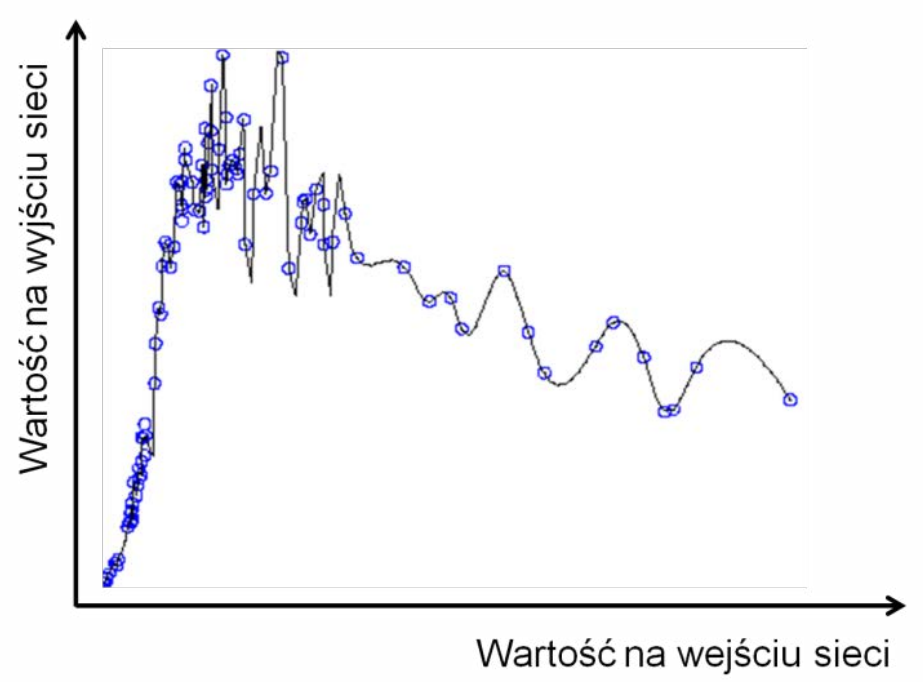
Przebieg genetycznej optymalizacji sieci

Wybór najwłaściwszej struktury sieci neuronowej może się odbywać za pomocą **algorytmu genetycznego**. Uzyskana tą drogą **genetyczna sieć neuronowa** miewa często lepsze właściwości niż sieć wymyślona przez użytkownika. Trzeba sobie jednak zdawać sprawę z tego, że przebieg genetycznej optymalizacji sieci może trwać długo i bywa bardzo burzliwy, gdyż w trakcie poszukiwania odpowiednio dobrej sieci, za sprawą czynników losowych, następują niekiedy gwałtowne polepszenia działania sieci, po których następują równie gwałtowne pogorszenia. Przedstawia to rysunek, na którym po lewej stronie pokazano schemat często stosowanego algorytmu genetycznego, a po prawej zmienność jakości sieci (wyrażanej przez popełniany przez nią sumaryczny **błąd**) w zależności od numeru populacji.



Przeuczenie

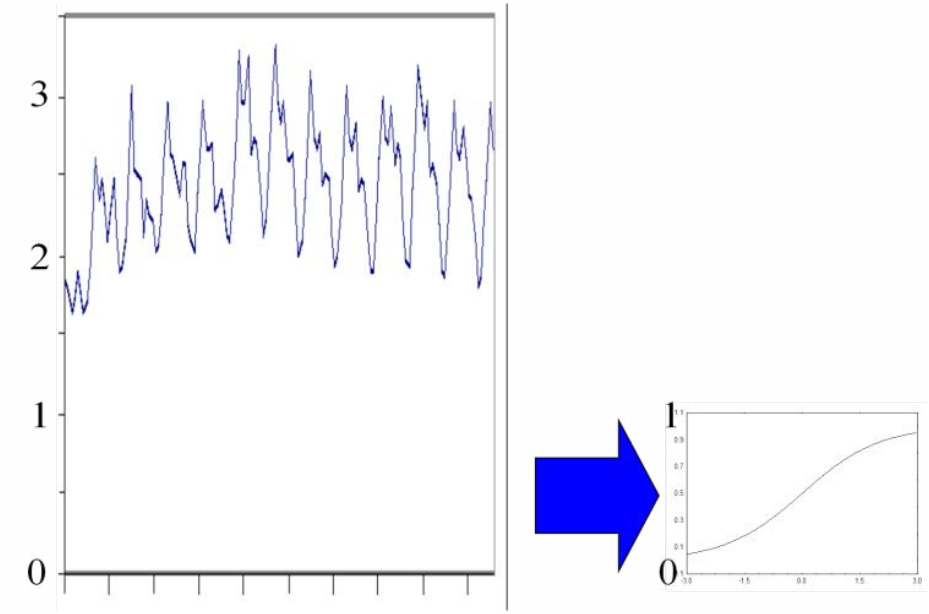
Zbyt długie **uczenie** sieci neuronowej powoduje, że sieć nadmiernie uzależnia swoje działanie od cech użytych do uczenia **przypadków uczących** – w tym także od cech drugorzędnych, nie dających podstaw do **generalizacji**.



Rysunek pokazuje przykład działania sieci przeuczonej. Działanie sieci (reprezentowane przez linię ciągłą na wykresie) zostało nadmiernie dopasowane do **przypadków uczących** (kółka na wykresie). Taka sieć będzie się cechowała bardzo złą **generalizacją**.

Przygotowanie ilościowych danych wejściowych dla sieci neuronowej

Sieć neuronowa jest systemem o dość dużej zdolności do dostosowywania swojego działania do różnych zadań i do różnych wymagań wynikających z tych zadań. Jednak we wszystkich tych zadaniach tylko dobre przygotowanie danych podawanych do **warstwy wejściowej** sieci gwarantuje sensowne użycie tego narzędzia i użyteczny wynik końcowy. Przykładem zagadnienia, które trzeba rozwiązać, jest dopasowanie przebiegu zmienności **danych wejściowych** do przedziału zmienności sygnałów mogących się pojawiać na wyjściach neuronów na skutek ich nieliniowej charakterystyki, co ilustruje podany niżej rysunek.



Jak widać z rysunku, dane wejściowe trzeba przesunąć (tzw. offset) i przeskalować, żeby dobrze odpowiadały wartościom występującym w sieci neuronowej, które w przypadku typowo używanej sigmoidalnej **funkcji aktywacji** przyjmują wartości wyłącznie z przedziału od 0 do 1.

Znacznie trudniejsze jest **przygotowanie jakościowych danych wejściowych dla sieci neuronowej**.

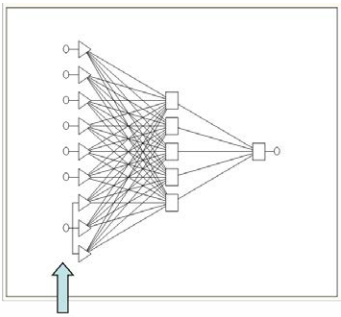
Przygotowanie jakościowych danych wejściowych dla sieci neuronowej

Dane wejściowe używane w **sieciach neuronowych** mogą mieć niekiedy formę danych *jakościowych*. Takie dane zamiast wartości liczbowych mogą przyjmować wartości w postaci *nazw*. Konkretna nazwa wskazuje jedną z możliwych wartości odpowiedniej danej, więc wnosi wartościową informację. Nie może ona jednak być bezpośrednio wykorzystywana w obliczeniach, bo nie jest liczbą. A powinna być uwzględniona!

Na przykład w zadaniu prognozowania przez sieć neuronową rynkowych cen mieszkań, większość danych wejściowych ma charakter ilościowy (powierzchnia mieszkania, wiek domu itp.), natomiast istotne informacje o tym, jak bardzo komfortowe jest mieszkanie albo w jakiej dzielnicy miasta jest położone – nie dadzą się wyrazić za pomocą liczb. Takie dane wejściowe przygotowuje się przed wprowadzeniem do sieci neuronowej, kodując je metodą określaną jako *„jeden-z-N"*. Kodowanie to polega na tym, że jeśli rozważana dana jakościowa może przyjmować jedną z *N* rozróżnialnych wartości, to dla jej reprezentacji na wejściu sieci neuronowej używa się *N* neuronów. Każdemu z tych neuronów przypisuje się jedną z możliwych wartości, jakie ta dana może przyjąć. Na przykład (patrz rysunek) jeśli dana dotyczy pochodzenia towaru, a miejscem pochodzenia może być Azja, Ameryka lub Europa, to odpowiednio pierwszy neuron sygnalizuje Azję, drugi Amerykę, a trzeci Europę. Gdy chcemy wprowadzić do sieci dane dotyczące na przykład pochodzenia towaru z Ameryki, to do pierwszego neuronu przesyłamy wartość 0 (bo nie jest to towar z Azji) do drugiego 1 (tak, to jest towar z Ameryki) i do trzeciego 0 (nie jest to wyrób europejski). Przy *N* możliwych kategoriach zawsze tylko na jedno **wejście** podajemy 1, a na wszystkie pozostałe 0.

Pochodzenie = {Azja, Ameryka, Europa}

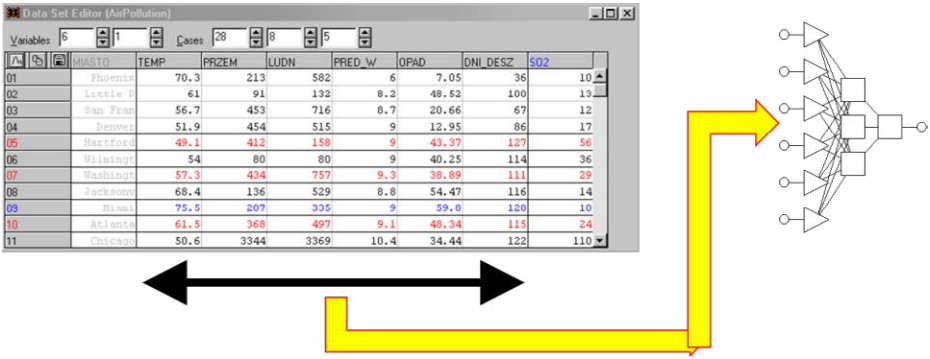
Azja:	*{1, 0, 0}*
Ameryka:	*{0, 1, 0}*
Europa:	*{0, 0, 1}*



jedna zmienna – trzy neurony!

Przykładowe dane wejściowe

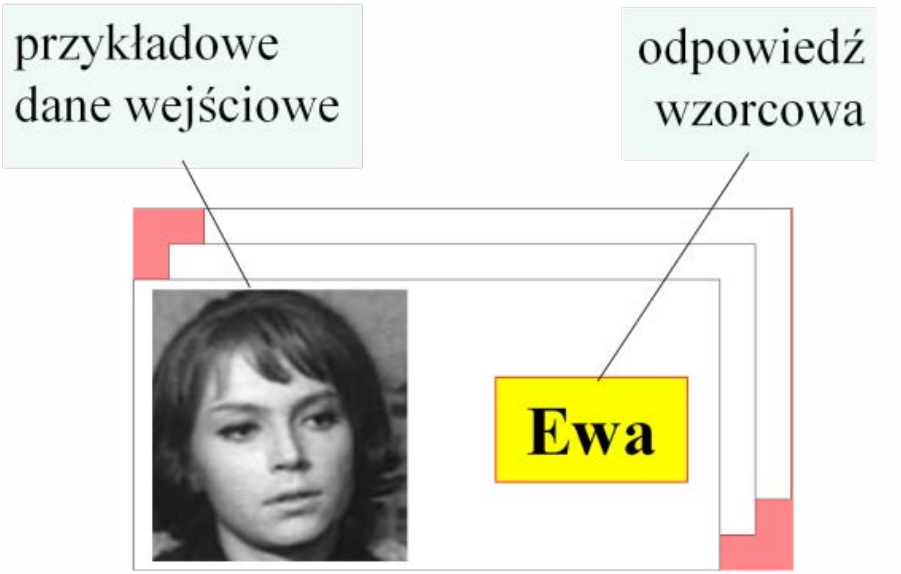
Pierwsza składowa każdego **przypadku uczącego** należącego do **zbioru uczącego**. Jest to komplet **danych wejściowych**, dla którego znany jest komplet poprawnych **danych wyjściowych** sieci (tak zwana **odpowiedź wzorcowa**).



Na rysunku przykładowe dane wejściowe wskazano zaznaczając strzałką odpowiedni zakres kolumn w typowym arkuszu, w którym przygotowywane bywają dane tworzące **zbiór uczący** dla sieci.

Przypadek uczący

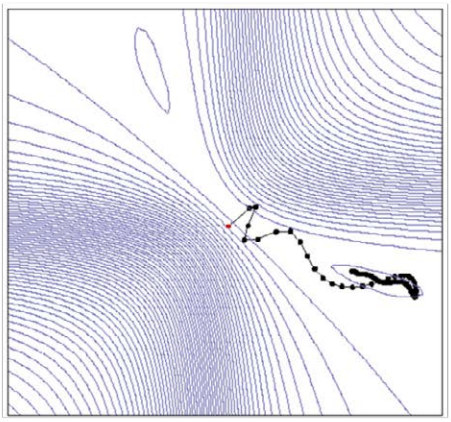
Jest to wzorzec poprawnie rozwiązanego zadania, którego powinna nauczyć się **sieć neuronowa**. Typowy przypadek uczący składa się z **przykładowych danych wejściowych** oraz z **odpowiedzi wzorcowej**, pokazującej komplet poprawnych **danych wyjściowych** dla **danych wejściowych** stanowiących pierwszą składową tego przypadku uczącego.



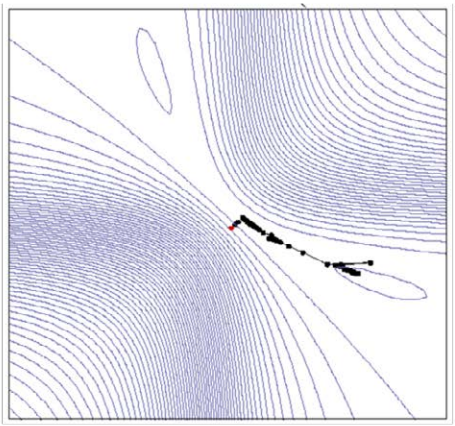
Na rysunku pokazano przypadek uczący dla hipotetycznej **sieci neuronowej**, która po pokazaniu zdjęcia rozpoznawanej osoby – powinna podać jej imię.

Quickpropagation

Odmiana algorytmu **wstecznej propagacji błędu**, która dzięki dostosowywaniu w poszczególnych krokach procesu **uczenia** wielkości **współczynnika uczenia** do lokalnych właściwości **funkcji błędu** pozwala na znaczne przyspieszenie procesu uczenia. Na rysunku pokazano przebieg uczenia sieci neuronowej przy pomocy podstawowego algorytmu **wstecznej propagacji błędu** oraz przy pomocy Quickpropagation. Widać, że w obu przypadkach sieć została prawidłowo nauczona. Osiągnięte zostało minimum funkcji błędu, której wartości przedstawiono na rysunku w postaci poziomic (jak na mapie geograficznej). Jednak widać, liczba iteracji wymaganych przy stosowanie algorytmu Quickpropagation była znacząco mniejsza.



Wsteczna propagacja błędów – 91 iteracji



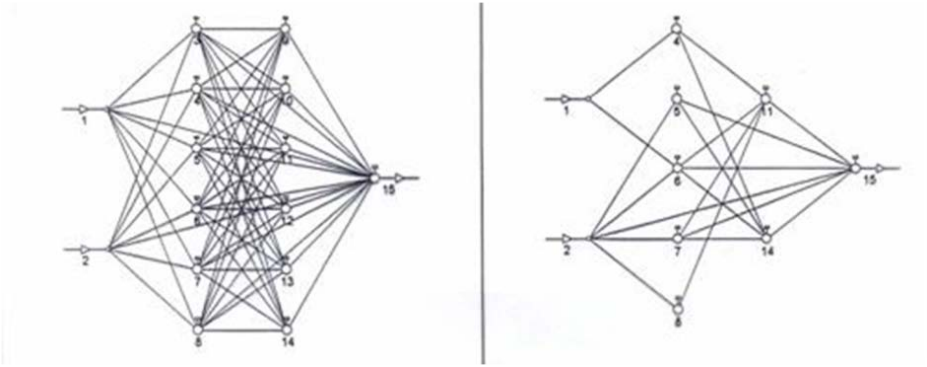
Algorytm Quickpropagation – 63 iteracje

RBF

Patrz hasło: **Sieć radialna**.

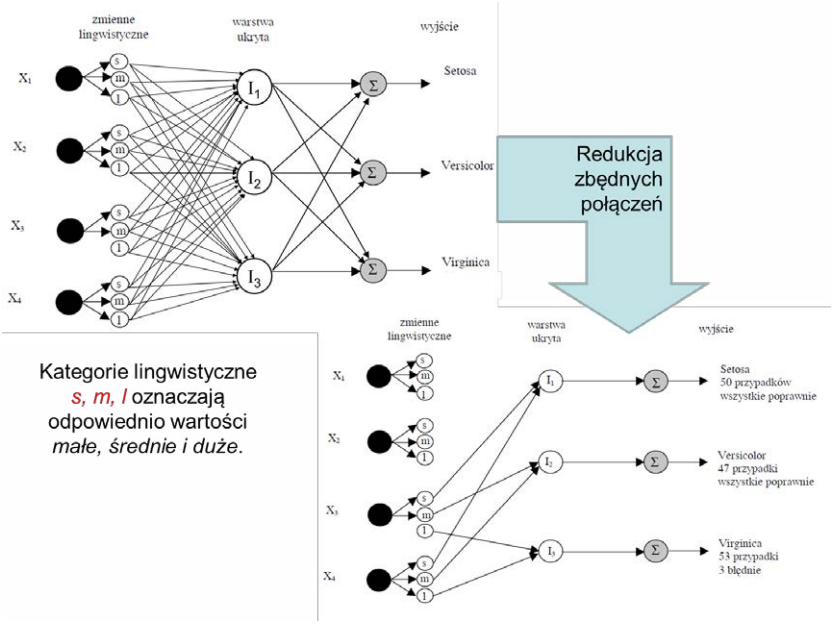
Redukcja połączeń

Projektując **strukturę sieci** z reguły przyjmuje się, że **neurony** sąsiednich **warstw** łączy się na zasadzie **każdy z każdym**. Jednak w procesie **uczenia** okazuje się, że wiele z tych **połączeń** jest niepotrzebnych. Przy prawidłowo prowadzonym **uczeniu** takie niepotrzebne połączenia otrzymują wartości **wag** wynoszące zero. W praktyce oznacza to, że połączenia te nie przesyłają **sygnałów** (dowolna wartość **sygnału** przemnożona przez zerową **wagę** daje zero). Jeśli **sieć** ma być po nauczeniu intensywnie eksploatowana – celowe jest przejrzenie jej **struktury** i usunięcie wszystkich takich „wyzerowanych" **połączeń**. Przy okazji redukcji niepotrzebnych **połączeń** usunąć można także niepotrzebne neurony – takie, do których nie docierają żadne niezerowe **sygnały wejściowe** oraz takie, których **sygnały wyjściowe** nie są przesyłane dalej w sieci (wyzerowane są wszystkie wagi połączeń wiodących z wyjść tych neuronów). O tym, jak bardzo taki zabieg może być efektywny, świadczy rysunek pokazujący po lewej stronie początkową **strukturę sieci**, a po prawej – to, co z niej pozostało po redukcji połączeń.



Redukcja warstwy wejściowej

Warstwa wejściowa sieci neuronowej nie powinna zawierać zbyt wielu neuronów. Problem ten jest omówiony w haśle **warstwa wejściowa**, natomiast tu jest przykład redukcji warstwy wejściowej, to znaczy zmniejszania liczby budujących ją elementów. Rozwiązywanym zadaniem jest klasyczny problem klasyfikacji trzech gatunków irysów[1] (*Setosa – Se, Versicolour – Ve i Virginica – Vi*) na podstawie wymiarów płatków składających się na ich kwiat. Ilustracja do tego problemu pokazana jest także w haśle **warstwa wejściowa**.

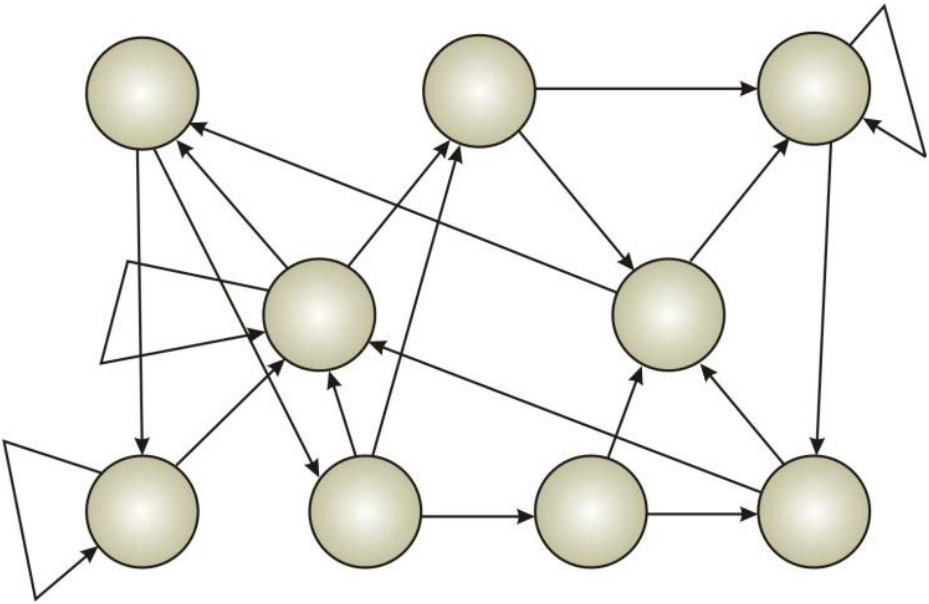


Na rysunku widać, że po **procesie uczenia** i po związanej z nim **redukcji połączeń** z 12 **danych wejściowych** pozostało jedynie 6, związanych z sygnałami X_3 i X_4. Prawidłowość tej redukcji danych jest wiarygodna, ponieważ wynik rozpoznawania w sieci po **redukcji połączeń** był zadowalający: 50 irysów gatunku *Se* rozpoznano poprawnie, 47 irysów gatunku *Ve* też rozpoznano poprawnie, a 3 omyłkowo zaliczono do gatunku *Vi*, zaś wszystkie irysy należące do gatunku Vi też rozpoznano bezbłędnie.

[1] Problem ten jest typowym *benchmarkiem* używanym w wielu testach. Dane do tego problemu wraz z opisem dostępne są na stronie http://archive.ics.uci.edu/ml/datasets/Iris

Rekurencyjna sieć neuronowa

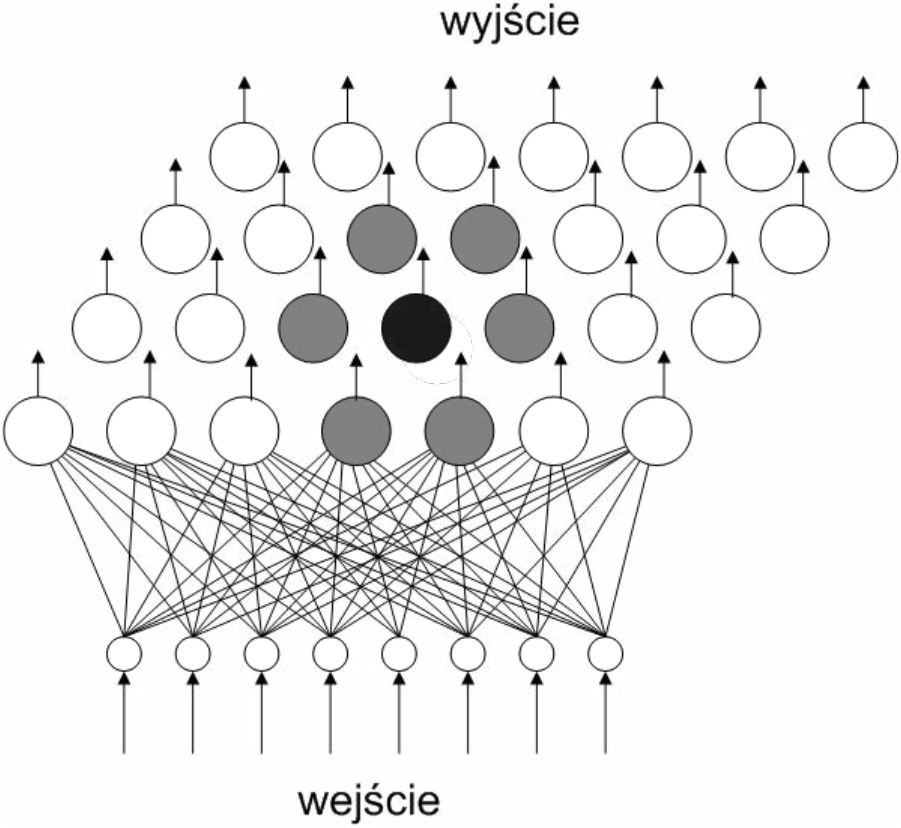
Struktura neuronalna, w której **sygnał** otrzymany na **wyjściu** sieci trafia powtórnie na jej **wejście** (taki obieg sygnału zwany jest sprzężeniem zwrotnym). Jednorazowe pobudzenie struktury ze sprzężeniem zwrotnym może generować całą sekwencję nowych zjawisk i sygnałów, ponieważ sygnały z wyjścia sieci trafiają ponownie na jej wejścia, generując nowe sygnały aż do ustabilizowania się sygnałów wyjściowych. Takiemu przebiegowi sygnałów wyjściowych wszystkich neuronów, powstającemu w sieci spontanicznie i nie poddającemu się żadnej kontroli, towarzyszą często oscylacje, gwałtowne narastanie sygnałów (wręcz do nieskończoności!) lub ich bezsensowne wygaszanie aż do całkowitego wyzerowania całej sieci. Szczególnie skomplikowane są procesy powstawania i rozwoju chaosu, z którymi w takiej sieci też często miewamy do czynienia.



Samoorganizująca się sieć neuronowa

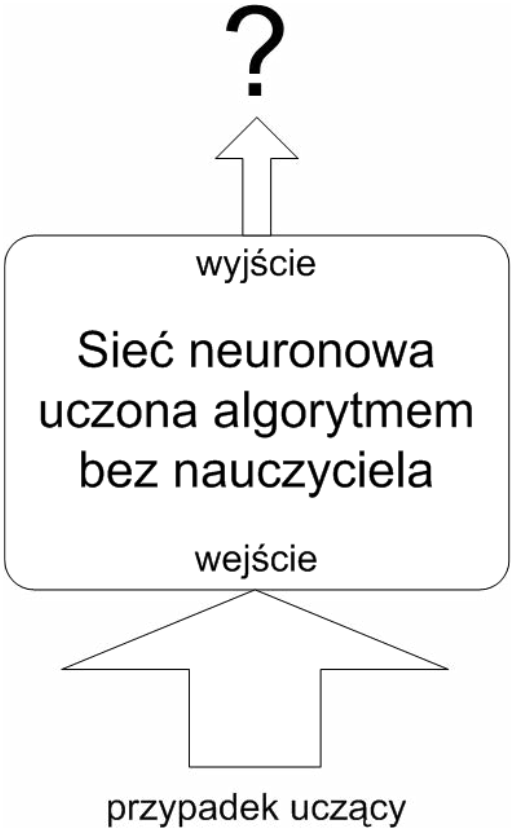
Samoorganizujące się sieci neuronowe określa się często jako **sieci neuronowe Kohonena**. Sieci tego typu realizują **samoorganizujące się odwzorowanie – SOM** (ang. *Kohonen's Self-Organizing Feature Map* – KSFM) i są ściśle związane ze sposobem ich uczenia, tzn. uczeniem nienadzorowanym, inaczej zwanym uczeniem bez nauczyciela lub **samouczeniem**. Oznacza to, że proces modyfikacji wag w tych sieciach odbywa się z użyciem zbioru uczącego, który zawiera jedynie informacje o wartościach **wejściowych** nieskojarzonych z żadnymi żądanymi wartościami **wyjściowymi**. Pozwala to odpowiednio dobranemu **algorytmowi samouczącemu** na w miarę swobodną interpretację informacji zawartej w **przypadkach uczących** znajdujących się w **zbiorze uczącym**. Dzięki temu możliwe staje się na przykład określenie liczby klas wektorów występujących w danym problemie i przypisanie każdej klasie reprezentanta w postaci neuronu w sieci. Jest to wspólna cecha sieci uczonych algorytmami bez nauczyciela, wśród których samoorganizujące się sieci neuronowe wyróżniają się procedurą modyfikacji wag w sieci.

W odróżnieniu od innych algorytmów, takich jak np. WTA (ang. *Winner Takes All* – zwycięzca bierze wszystko), w samoorganizującej się sieci neuronowej w trakcie prezentacji danego **przypadku uczącego** modyfikowane są **wagi** nie tylko **neuronu** reprezentującego klasę, do której przyporządkowuje się ten przypadek, ale również **wagi neuronów** znajdujących się w topologicznym **sąsiedztwie** tego **neuronu**. Pozwala to na odwzorowanie rozkładu klas w przestrzeni **wektorów wejściowych** na topologiczny rozkład **neuronów** reprezentujących te klasy, na zasadzie: podobne do siebie klasy reprezentowane są przez bliskie sobie **neurony**. **Sąsiedztwo** topologiczne może być realizowane w jednym lub częściej dwóch wymiarach, co determinuje jedno lub dwuwymiarowość warstwy neuronów w samoorganizującej się sieci.


wyjście
wejście

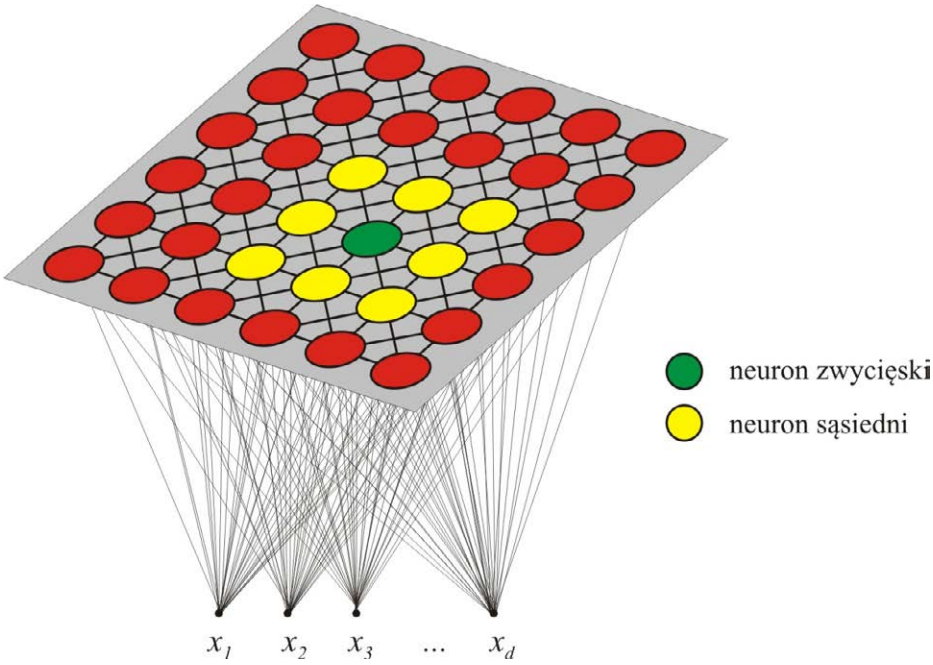
Samouczenie

Samouczenie, inaczej zwane uczeniem nienadzorowanym lub uczeniem bez nauczyciela, jest odmianą uczenia **sieci neuronowych**, w którym procesy modyfikacji wartości **wag** w tej sieci przeprowadzane są z użyciem zbioru uczącego, który nie zawiera informacji w postaci wartości żądanych na **wyjściach** sieci przy wartościach reprezentujących **przypadki uczące** podanych na **wejściach** sieci. Odpowiednio sformułowany proces samouczenia bazujący na tak określonym zbiorze uczącym pozwala na swobodną, w pewnym zakresie, interpretację informacji zawartej w **przypadkach uczących**. Samouczenie pozwala **sieciom neuronowym** na rozwiązywanie problemów związanych na przykład z sytuacją, gdy brak jest przesłanek do wyrokowania o liczbie i rozkładzie klas w analizowanym problemie klasyfikacyjnym. W tym przypadku mówi się o możliwości rozwiązywania przez **sieć neuronową** problemu grupowania danych. Najbardziej popularne **algorytmy realizujące samouczenie** to: algorytm określany jako WTA (ang. *Winner Takes All* – zwycięzca bierze wszystko) oraz algorytm KSFM (ang. *Kohonen's Self-Organizing Feature Map* – **somoorganizujące się odwzorowanie** Kohonena). W przypadku zastosowania pierwszego z tych algorytmów, po przeprowadzeniu samouczenia, uzyskuje się informację o szacunkowej liczbie występujących w danym praktycznym problemie klas oraz o przybliżonych wartościach środków ciężkości tych klas, czyli określa się reprezentantów poszczególnych klas w postaci wybranych neuronów W przypadku bardziej złożonego algorytmu Kohonena, oprócz powyższych danych, ustalane jest topologiczne rozmieszczenie reprezentantów określonych klas w warstwie neuronów na zasadzie: **neurony** reprezentujące podobne do siebie klasy umieszczane są w swoim **sąsiedztwie** w strukturze sieci neuronowej.


?
wyjście
Sieć neuronowa uczona algorytmem bez nauczyciela
wejście
przypadek uczący

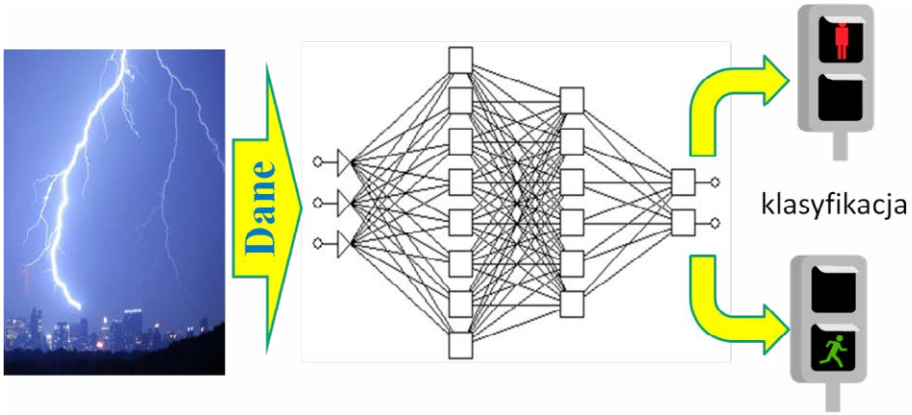
Sąsiedztwo

W większości sieci neuronowych nie rozważa się tego, jak neurony są rozmieszczone, gdyż nie ma to żadnego znaczenia. Wyjątkiem są **sieci Kohonena**, znane też jako **samoorganizujące się sieci neuronowe**. W warstwie topologicznej tych sieci położenie neuronów ma znaczenie, gdyż neurony położone w pobliżu zwycięskiego neuronu (patrz hasło **Konkurencyjna sieć neuronowa**), czyli będące jego sąsiadami, są odmiennie traktowane w trakcie procesu uczenia oraz mają specjalne znaczenie podczas interpretacji wyników dostarczanych przez pracującą (nauczoną) sieć.



Sieć klasyfikacyjna

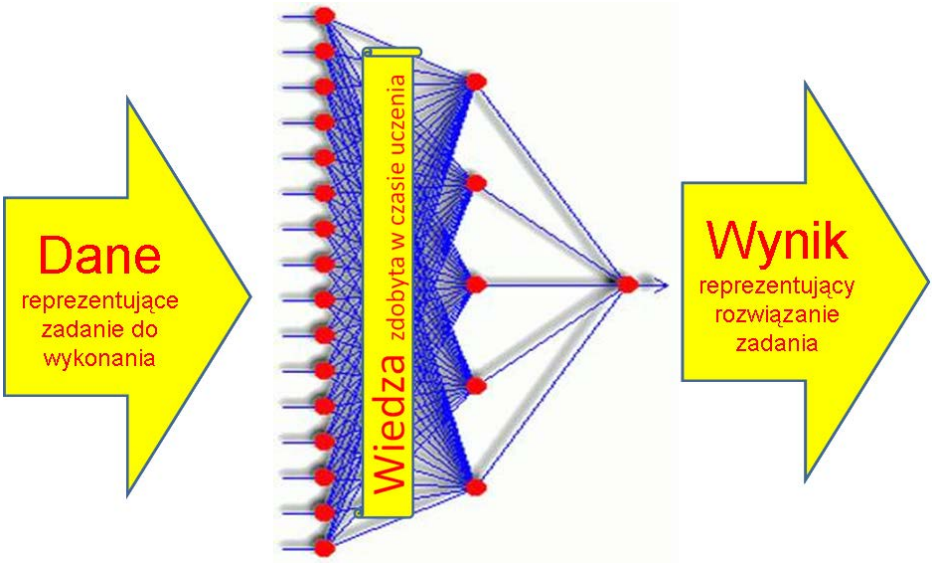
Odmiana **sieci neuronowej**, w której sygnały wyjściowe mają charakter jakościowy (pojawiają się na **wyjściu** jako liczby, najczęściej z przedziału [0,1], ale są wykorzystywane w formie decyzji). Należy zwrócić uwagę, że każdej z przewidywanych klas odpowiada jeden neuron w warstwie wyjściowej.



Sieć neuronowa

System przeznaczony do przetwarzania informacji, którego budowa i zasada działania są w pewnym stopniu wzorowane na funkcjonowaniu fragmentów rzeczywistego (biologicznego) systemu nerwowego. Na przesłankach biologicznych oparte są schematy sztucznych **neuronów** wchodzących w skład sieci oraz (w pewnym stopniu) jej **struktura**. Jednak schematy **połączeń neuronów** w sieci neuronowej są wybierane arbitralnie, a nie stanowią modelu rzeczywistych struktur nerwowych.

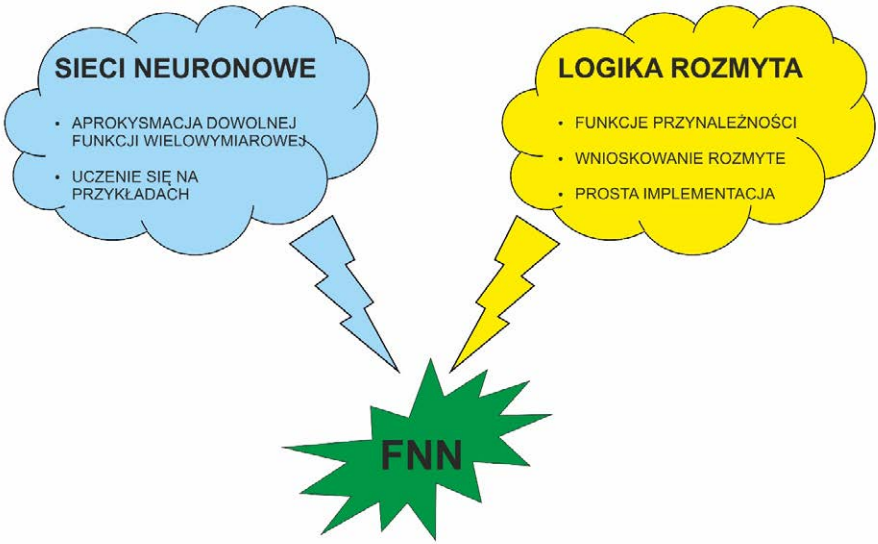
Wyróżniającą cechą sieci neuronowej jako narzędzia informatycznego jest możliwość komputerowego rozwiązywania przy jej pomocy praktycznych problemów bez ich uprzedniej matematycznej formalizacji. Dalszą zaletą jest brak konieczności odwoływania się przy stosowaniu sieci do jakichkolwiek teoretycznych założeń na temat rozwiązywanego problemu. Nawet założenie o przyczynowo-skutkowych zależnościach między **wejściem** a **wyjściem** nie musi być egzekwowane! Najbardziej znamienną cechą sieci neuronowej jest jej zdolność **uczenia** się na podstawie przykładów i możliwość automatycznego uogólniania zdobytej wiedzy (**generalizacja**). Rysunek przedstawia najbardziej typowy schemat praktycznego użycia sieci neuronowej.



Sieć neuronowo-rozmyta

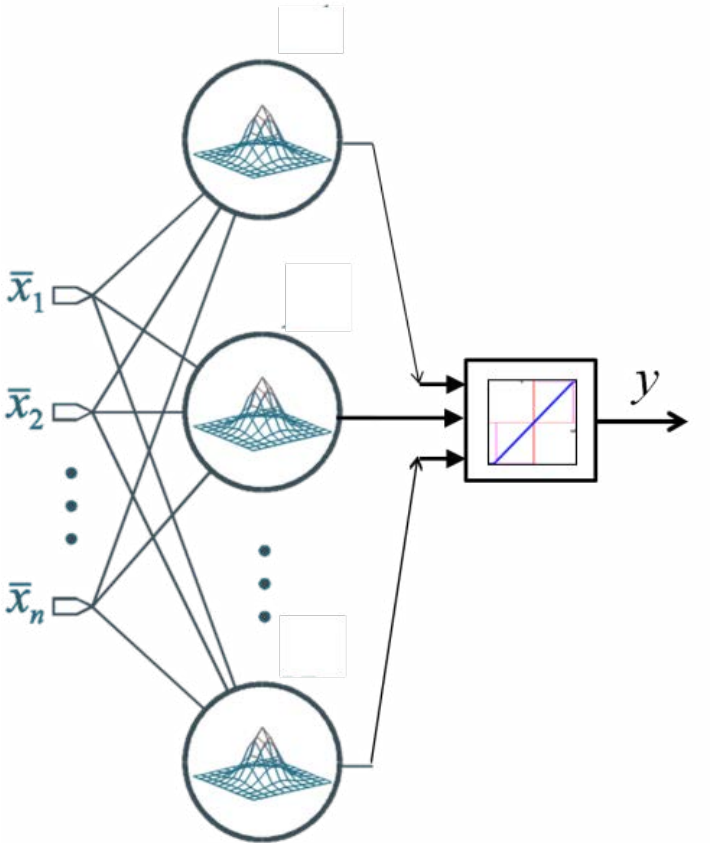
Jest to realizacja rozmytego systemu decyzyjnego w postaci sieci (na wzór **sieci neuronowej**) elementów realizujących różnego rodzaju funkcje i operacje elementarne. Struktura sieciowa pozwala na implementację procesu optymalizacji w oparciu o kryterium błędu średniokwadratowego z wykorzystaniem metody największego spadku – analogicznej do **wstecznej propagacji błędów** stosowanej w przypadku jednokierunkowych **sieci neuronowych**. Rozwiązanie to pozwala połączyć zalety rozmytych systemów decyzyjnych (czytelność wiedzy) i sieci neuronowych (możliwość uczenia).

Dzięki strukturze sieciowej możliwe jest przeprowadzenie optymalizacji parametrów zbiorów rozmytych w odniesieniu do prawidłowego odwzorowania próbek wzorcowych zawartych w **zbiorze uczącym**. Tak jak w przypadku **wstecznej propagacji błędów** stosowanej w odniesieniu do **sieci neuronowych**, również w przypadku sieci neuronowo-rozmytej wyznacza się **błąd** na wyjściu, a następnie propaguje go wstecz, prowadząc do wyznaczenia poprawek parametrów zbiorów rozmytych. Odmienne, w stosunku do **wstecznej propagacji błędów** stosowanej w **sieciach neuronowych**, są funkcje realizowane przez elementy, z których zbudowana jest sieć oraz pochodne tych funkcji.



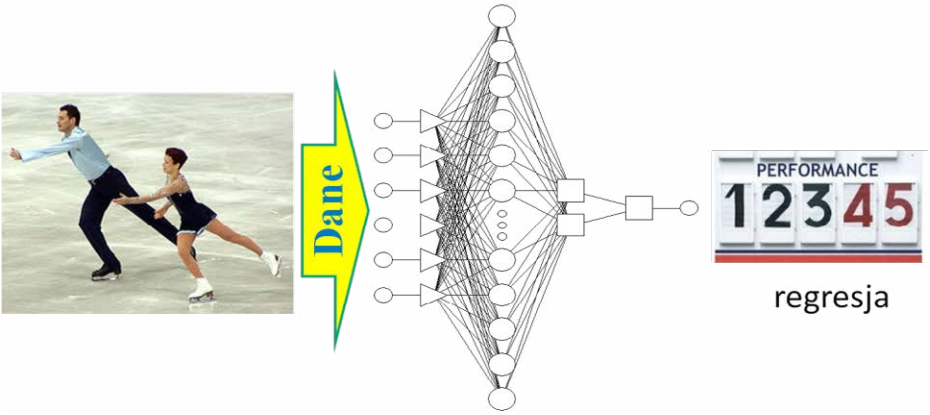
Sieć radialna

Rodzaj **jednokierunkowej sieci neuronowej** w której wykorzystywana jest technika radialnych funkcji bazowych (**RBF** – *Radial Basis Functions*) i stosowane są **neurony radialne**. Sieć radialna w typowym kształcie składa się (patrz rysunek) z **warstwy wejściowej** (jak zawsze nieangażowanej bezpośrednio w procesy przetwarzania informacji), **warstwy ukrytej** złożonej z **neuronów radialnych** (jest ich zwykle bardzo dużo) i **warstwy wyjściowej**, wypracowującej **odpowiedź sieci.** Neurony radialne służą do rozpoznawania powtarzalnych i charakterystycznych cech grup (skupisk) danych wejściowych. Konkretny neuron radialny ulega pobudzeniu, gdy sieć radialna konfrontowana jest z przypadkiem podobnym do tego, który nauczył się on wcześniej rozpoznawać jako reprezentanta pewnej grupy. W warstwie wyjściowej sieci radialnej najczęściej występuje (jak na rysunku) jeden **neuron liniowy** – ale bywają wyjątki od tej reguły.



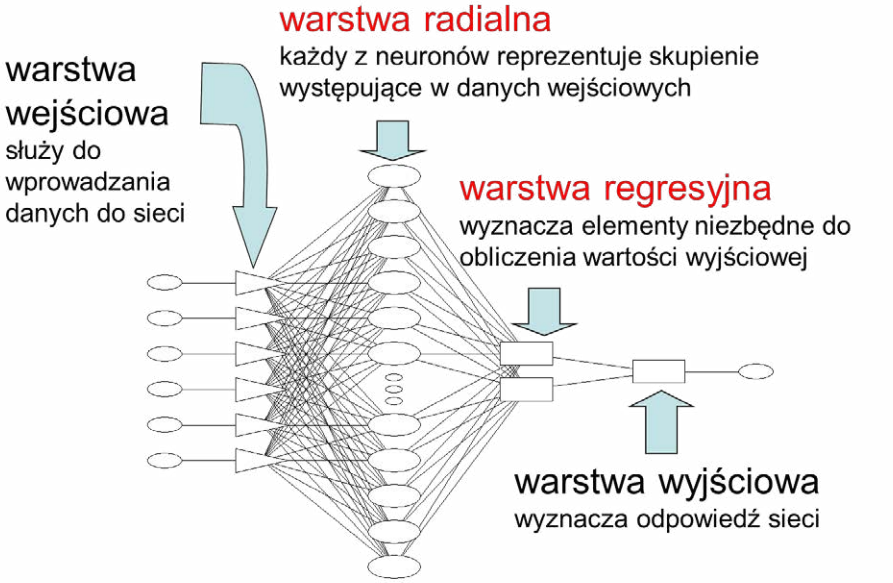
Sieć regresyjna

Odmiana **sieci neuronowej**, w której **sygnały wyjściowe** mają charakter ilościowy (dane pojawiające się na **wyjściu** są wykorzystywane w formie wartości liczbowych). Jest to jedna z częściej wykorzystywanych w praktyce form **sieci neuronowej**.



Sieć uogólnionej regresji

Sieć tego typu łączy zalety sieci radialnej i sieci MLP. Jest zwykle określana skrótem **GRNN** (od ang. *Generalized Regression Neural Network*). Jej budowa wraz ze wskazaniem roli poszczególnych warstw przedstawiona jest na rysunku poniżej.



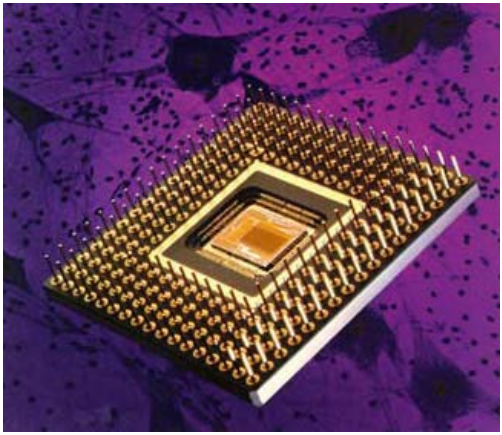
W warstwie radialnej, będącej tu odpowiednikiem pierwszej warstwy ukrytej, wykorzystywane są **neurony radialne**, dokonujące grupowania danych wejściowych. Warstwa ta może się składać z bardzo dużej liczby neuronów, co odpowiada wykryciu w zbiorze danych wejściowych dużej liczby skupień danych (patrz – **sieć radialna**). Druga warstwa składa się tylko z dwóch neuronów sumujących (tzw. **neuronu mianownikowego** i **neuronu licznikowego**) i jest nazywana warstwą regresyjną. Neuron **wyjściowy** wykonuje tylko jedno działanie, w wyniku którego powstaje iloraz wyników obu neuronów sumujących. Można wykazać, że w sieci GRNN uzyskuje się najlepszą estymację wymaganej wartości wyjściowej w **sieciach regresyjnych**.

SOM – samoorganizujące się odwzorowanie

Patrz hasło: **Samoorganizująca się sieć neuronowa**.

Sprzętowe realizacje

Większość zastosowań **sieci neuronowych**, które są opisywane w literaturze, a w szczególności wszystkie te zastosowania, które opisywane są we wcześniejszych rozdziałach tej monografii – dotyczą w istocie wykorzystania **programów modelujących sieci neuronowe**. Niemniej warto wspomnieć, że sieci neuronowe były i są niekiedy używane jako sprzętowe realizacje. Dawniej najczęściej były one wykonywane jako specjalizowane układy scalone ASIC (*Application Specific Integrated Circuit*), a obecnie głównie realizowane na bazie programowalnych układów FPGA (*Field Programmable Gate Array*). Na rysunku przedstawiono najbardziej znany przykład sprzętowej realizacji **sieci neuronowej** wykonanej przez firmę Intel. Obraz zaczerpnięty 15 lutego 2014 roku ze strony http://www.warthman.com/images/Intel%2080170%20B.jpg

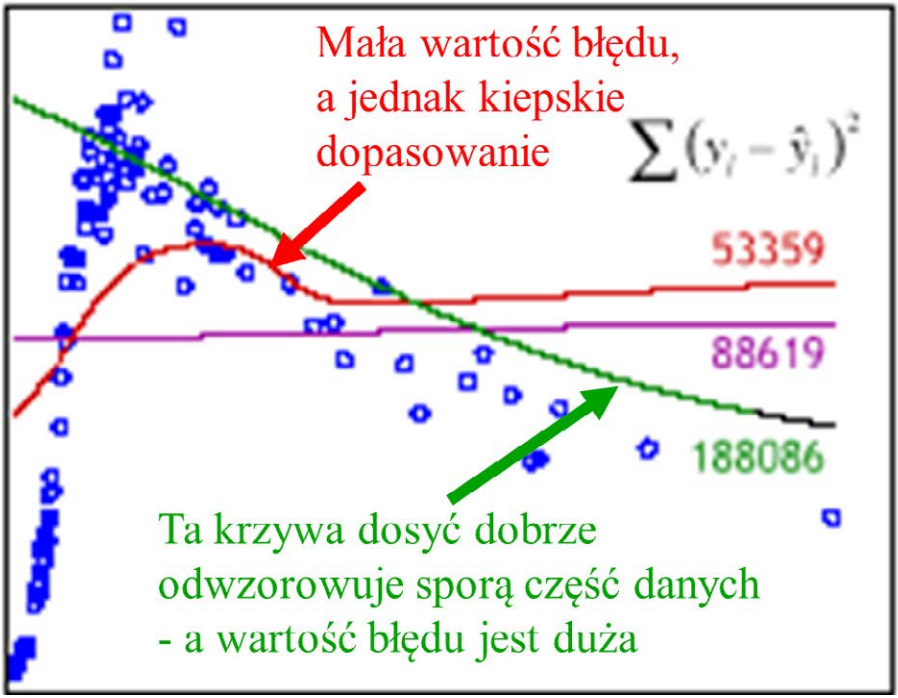


SSE – suma kwadratów błędów

W trakcie procesu uczenia sieci trzeba obserwować postęp tego procesu, ponieważ istnieje ryzyko, że nie przyniesie on pożądanego rozwiązania. Najwygodniej jest to zrobić obserwując, jak zmienia się wartość **błędu** w kolejnych **epokach** procesu uczenia. Ponieważ sieć może mieć wiele **wyjść** (powiedzmy, że jest ich *M*), a **epoka** składa się z *R* **przypadków uczących** – trzeba brać pod uwagę błąd całościowy, sumowany po wszystkich przypadkach uczących i po wszystkich wyjściach sieci. Zwykle przed zsumowaniem wartości błędów podnoszone są do kwadratu, żeby uniknąć efektu kompensowania błędów ujemnych przez błędy dodatnie, a ponadto operacja podnoszenia do kwadratu powoduje silniejsze zaakcentowanie dużych błędów przy równoczesnym zmniejszeniu wpływu błędów małych. Powstający wskaźnik nazywany jest SSE (*Sum Square Errors*) i jest wyrażany wzorem:

$$SSE = \sum_{p=1}^{R} \sum_{k=1}^{M} \left(d_{pk} - y_{pk}\right)^2$$

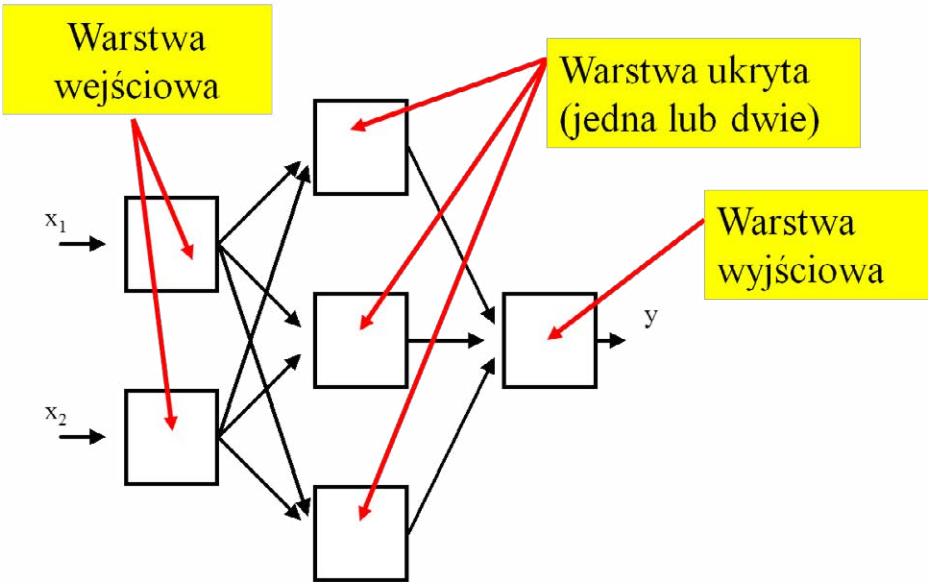
gdzie d_{pk} oznacza wzorcową odpowiedź, jaka powinna się pojawić przy prezentacji **przypadku uczącego** o numerze *p* na **wyjściu** sieci o numerze *k*, a y_{pk} oznacza wartość, jaka się w rzeczywistości pojawiła na tym wyjściu



SSE jest powszechnie stosowaną w technice sieci neuronowych miarą błędu popełnianego przez sieć i większość **algorytmów uczenia** sieci jest dostosowanych do jego minimalizacji. Jednak, jak pokazano na rysunku, nie zawsze mniejsza wartość tej miary (obrazują ją liczby obok odpowiednich wykresów) oznacza lepsze dopasowanie działania sieci do empirycznych danych, które powinna ona odwzorować.

Struktura sieci neuronowej

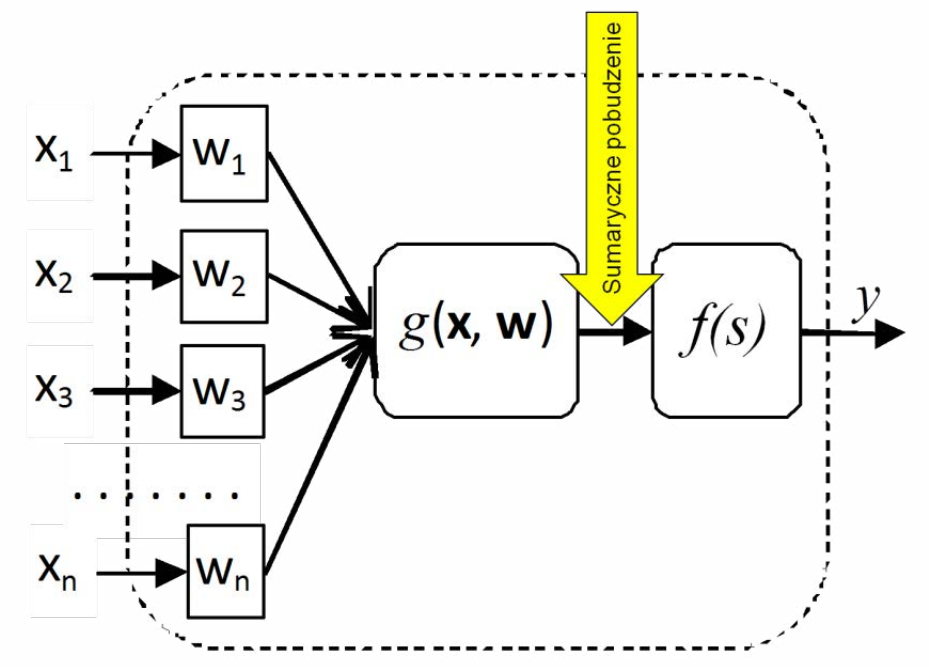
Struktura sieci neuronowej to sposób ułożenia i połączenia **neuronów**. W typowej sieci struktura zakłada istnienie **warstw**, bo takie ułożenie neuronów ułatwia definiowanie struktury sieci (gdy użytkownik musi zdecydować, jakiej sieci chce użyć), a także porządkuje sterowanie pracą sieci zarówno przy jej realizacji w formie programu symulującego sieć w zwykłym komputerze, jak i w przypadku stosowania specjalizowanych rozwiązań sprzętowych (chipów neuronowych, neurokomputerów itp.).



Zwykle w strukturze sieci wyróżnić można **warstwę wejściową**, **warstwę wyjściową** oraz **warstwy ukryte**. **Liczba warstw ukrytych** waha się od zera (może być sieć *bez* warstw ukrytych) do dwóch. Sieci o większej liczbie warstw ukrytych są inteligentniejsze (potrafią rozwiązywać trudniejsze zadania), ale są trudniejsze do **uczenia**.

Sumaryczne pobudzenie

W strukturze każdego **neuronu** musi być przewidziana czynność **agregacji danych wejściowych**, ponieważ jedną z zasadniczych cech **neuronu** jest to, że ma on wiele **wejść** i tylko jedno **wyjście**. Różne metody **agregacji** są omówione w oddzielnym haśle tego Leksykonu, natomiast tutaj podkreślony jest fakt, że wynik tej **agregacji** jest sygnałem skalarnym, który jest następnie użyty jako argument **funkcji aktywacji**. Właśnie ten skalarny sygnał nazywany jest sumarycznym pobudzeniem.



Surowość nauczyciela

Inspirująca i użyteczna analogia, pozwalająca lepiej zrozumieć rolę **współczynnika uczenia** w procesie **uczenia** sieci neuronowej. Jedną z interpretacji surowości nauczyciela (nauczającego dzieci) oraz jej związku z wartością **współczynnika uczenia** sieci neuronowej przedstawia rysunek poniżej.

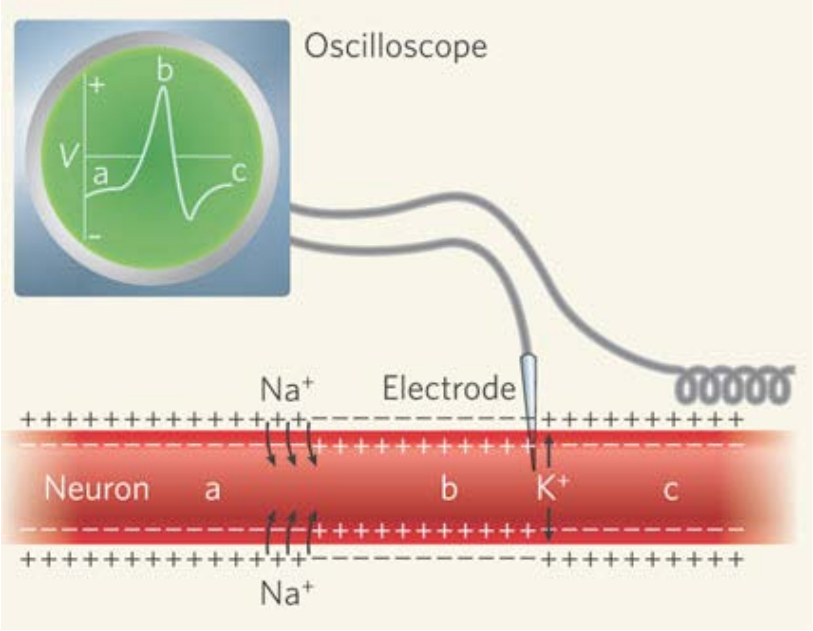
Duży współczynnik uczenia działa jak surowy nauczyciel zmuszający ucznia po każdym błędzie do radykalnej zmiany postępowania (duże korekty wag)

Mały współczynnik uczenia działa jak łagodny nauczyciel, akceptujący mały wpływ nawet kiepskich doraźnych ocen na działania wymuszające postępy w nauce

Jednym z wniosków (użytecznych!), jakie można wysnuć na podstawie analogii między współczynnikiem uczenia a surowością nauczyciela, jest propozycja **zmiany wartości współczynnika uczenia** w trakcie nauki.

Sygnały

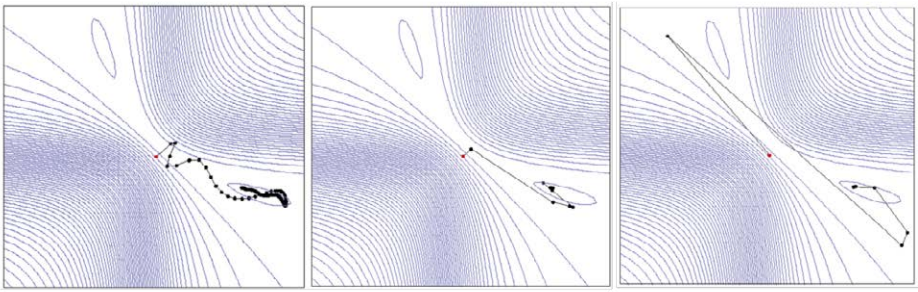
Dane, którymi operujemy w **sieciach neuronowych**, często nazywamy sygnałami, chociaż w istocie są to wartości liczbowe uczestniczące w określonych obliczeniach. Dotyczy to zarówno **danych wejściowych**, jak i **danych wyjściowych**. Nazewnictwo to bardzo się zakorzeniło w środowiskach często używających **sieci neuronowych**, jak również wśród badaczy tworzących nowe sieci i doskonalących ich działanie. Wynika to z faktu, że neurobiolodzy badający rzeczywiste komórki nerwowe obserwują zachodzące w nich procesy, korzystając z rejestracji różnych sygnałów bioelektrycznych (patrz rysunek poniżej zaczerpnięty 15 lutego 2014 ze strony http://people.eku.edu/ritchisong/301images/Neuron_action_potential.jpg), więc terminologia używana w ich publikacjach właśnie do tych sygnałów nawiązuje.



Z kolei twórcy **sieci neuronowych** wprawdzie obecnie zadowalają się **programami modelującymi sieci neuronowe**, ale mają także na względzie ich **sprzętowe realizacje**, w których znowu w użyciu są określone sygnały – tyle tylko, że czysto elektroniczne. Stąd częste używanie w literaturze dotyczącej sieci neuronowych (a także w tym *Leksykonie*) określenia 'sygnały' (**wejściowe**, **wyjściowe** i wewnętrzne) jako synonimu odpowiednich danych (**wejściowych**, **wyjściowych** i wewnętrznych).

Szybkie algorytmy uczenia

Metoda **wstecznej propagacji błędów** jest stosunkowo prymitywna, co powoduje, że **uczenie** sieci jest stosunkowo powolne. Pewne przyspieszenie procesu uczenia można uzyskać, stosując metodę **Quickpropagation**, ale przyspieszenie to nie jest szczególnie istotne. Natomiast istnieją metody uczenia, których zastosowanie może bardzo znacząco przyspieszyć uczenie sieci – chociaż stosowalność tych metod uzależniona jest od tego, czy **funkcja błędu** spełnia pewne dodatkowe warunki. Ponieważ zwykle nie da się sprawdzić, czy funkcja błędu te warunku spełnia, stosowanie tych przyspieszonych metod uczenia obarczone jest pewnym ryzykiem (przy niespełnionych warunkach **proces uczenia** może nie być zbieżny). Ale gdy metody te można zastosować – stopień przyspieszenia **procesu uczenia** jest bardzo duży. Przykładem takich metod jest algorytm gradientów sprzężonych, quasi-Newtona czy Leveberga-Marquardta (patrz rysunek, a także hasło **Quickpropagation**)



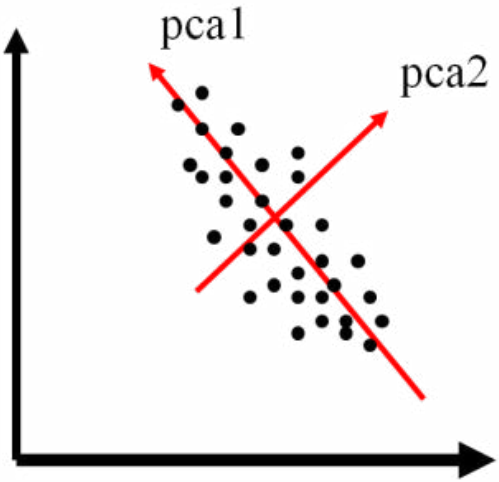
Wsteczna propagacja błędów – 91 iteracji

Metoda gradientów sprzężonych – 22 iteracje

Algorytm Levenberga-Marquardta – 7 iteracji

Transformacja PCA

Transformacja ta służy do przekształcenia **danych wejściowych** mających postać wektora o wielu składowych do takiej postaci, w której tę samą (prawie) ilość użytecznej informacji daje się uzyskać z wykorzystaniem wektora o znacznie mniejszej liczbie składowych (czyli w przestrzeni o mniejszym wymiarze). W klasycznej PCA uzyskuje się dodatkowo to, że składowe wektora po transformacji są zdekorelowane (ich wzajemne korelacje są zerowe), a składowe są uporządkowane według stopnia ważności – pierwsza składowa ma największą wartość (wnosi najwięcej informacji), następna nieco mniej itd. Ostatnie składowe wnoszą bardzo mało informacji, więc mogą zostać pominięte, co powoduje, że PCA zastosowana jako przetwarzanie wstępne sygnału przed jego wprowadzeniem do sieci neuronowej pozwala zmniejszyć rozmiar **warstwy wejściowej sieci**.

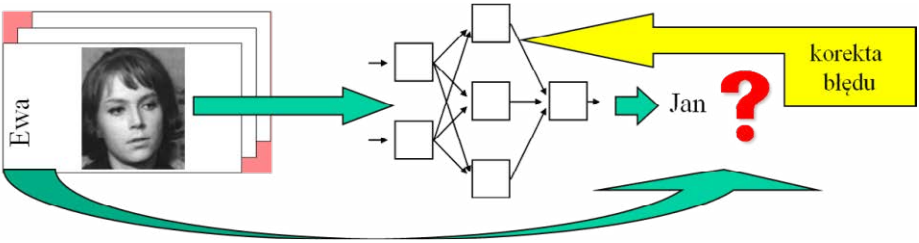


Rysunek przedstawia przykładowy zbiór **danych wejściowych** w oryginalnym układzie współrzędnych z wprowadzonym układem współrzędnych PCA. Widać, że współrzędna *pca1* opisuje zmienność rozważanych danych na tyle dobrze, że wprowadzanie współrzędnej *pca2* nie jest konieczne. W ten sposób problem dwuwymiarowy można zredukować do jednowymiarowego.

Obok klasycznej PCA stosowana jest także nieliniowa PCA. Nieliniową PCA uzyskuje się z wykorzystaniem **sieci autoasocjacyjnej**.

Uczenie

Uczenie jest to proces oparty na prezentacji **przypadków uczących** (przykładów prawidłowo rozwiązanych zadań) należących do **zbioru uczącego**. W trakcie tych pokazów następuje stopniowe dopasowywanie się sieci do tego, by nabyła ona umiejętność rozwiązywania tych zadań. Dopasowywanie to opiera się na porównywaniu **odpowiedzi** udzielanych przez sieć z **odpowiedziami wzorcowymi**. Wprowadzana korekta błędu powoduje, że sieć po każdej prezentacji zwiększa szansę udzielenia **odpowiedzi** bardziej zbliżonej do **odpowiedzi wzorcowej**, a ponadto zmierza się do tego, żeby sieć uogólniła wyuczoną umiejętność na inne, podobne zadania, nieprezentowane w trakcie uczenia (**generalizacja**).

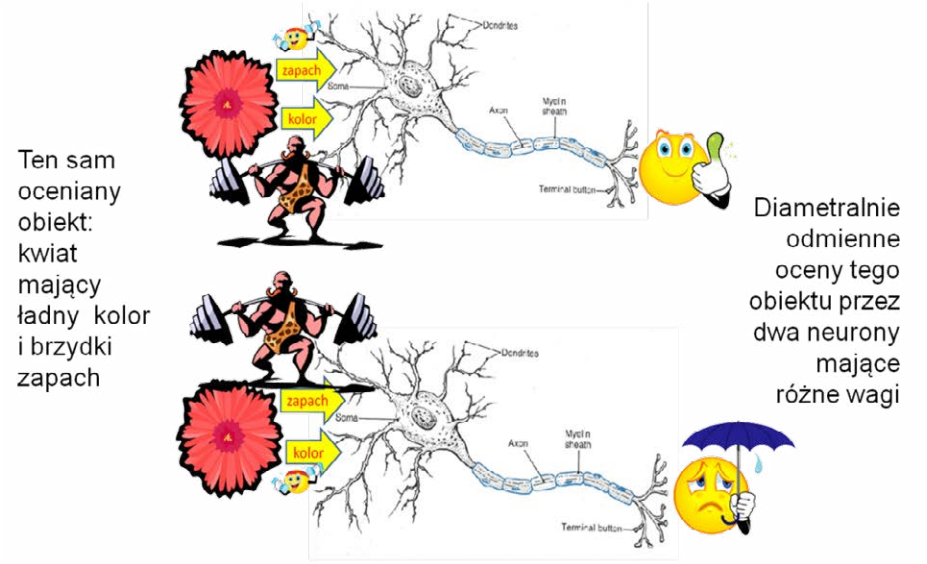


Na rysunku pokazano schemat uczenia dla hipotetycznej sieci neuronowej, która po pokazaniu zdjęcia rozpoznawanej osoby – powinna podać jej imię.

Formalnie **uczenie** można określić jako iteracyjny proces estymacji optymalnych wartości parametrów sieci (najczęściej **wag**) na podstawie **zbioru uczącego**. Metodę zmiany **wag** w trakcie procesu uczenia wyznacza używany **algorytm uczenia**.

Wagi

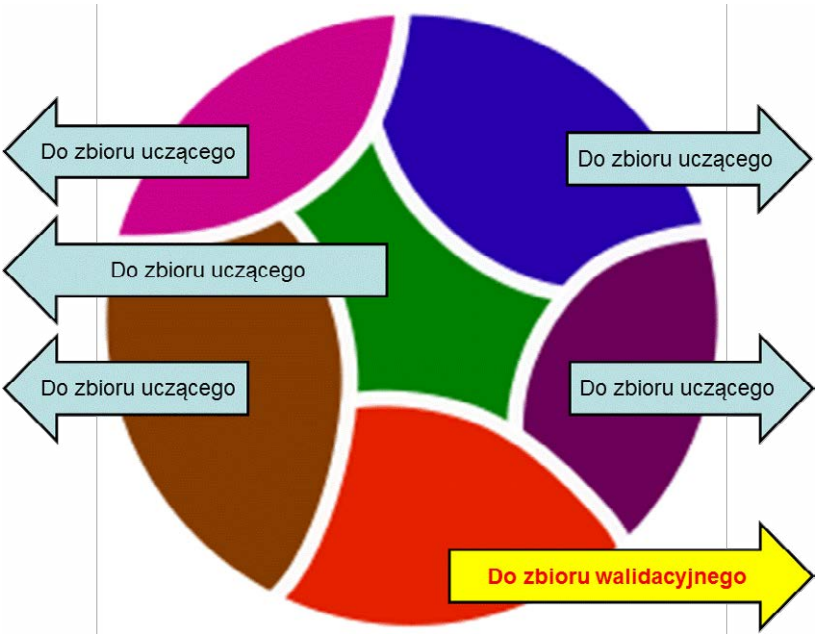
Parametry **neuronu** decydujące o jego właściwościach i roli w procesie rozwiązywania przez sieć postawionego zadania. Zwykle wagi dopasowuje w całej sieci używany **algorytm uczenia** lub samouczenia. Komplet wartości wag ustalonych we wszystkich **neuronach** w trakcie **uczenia** lub **samouczenia** determinuje wiedzę, jaką posiada sieć neuronowa.



Jak pokazano na rysunku – od przypisania do różnych **wejść** neuronu różnych wag zależy jego zachowanie.

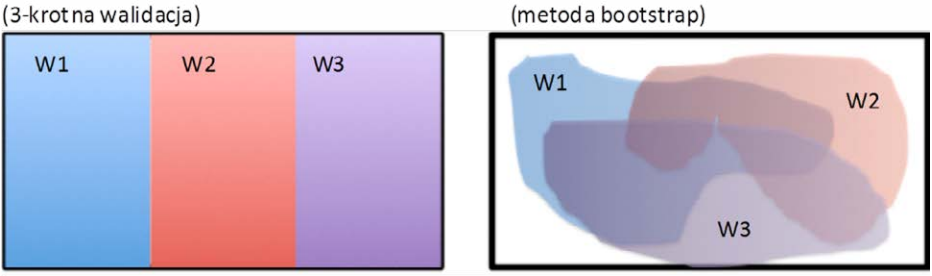
Walidacja krzyżowa

Do oceny zdolności sieci do **generalizacji** można wykorzystać metodę określaną jako *walidację krzyżową*. Metoda ta polega na podziale zbioru danych wejściowych sieci na szereg podzbiorów, pośród których wybiera się jeden z nich jako zbiór walidacyjny, natomiast pozostałe podzbiory służą do uczenia sieci (tworzą **zbiór uczący**). Metoda ta jest stosowana szczególnie wtedy, kiedy liczba danych wejściowych jest ograniczona. Przeciwieństwem walidacji krzyżowej jest walidacja prosta, kiedy w sposób losowy tworzy się tylko dwa podzbiory: **uczący** i **walidacyjny**. Metoda walidacji krzyżowej posiada kilka odmian. Najpopularniejszymi są: **walidacja *n*-krotna**, **walidacja metodą *leave-one-out*** oraz **walidacja metodą *bootstrap***.



Walidacja metodą *bootstrap*

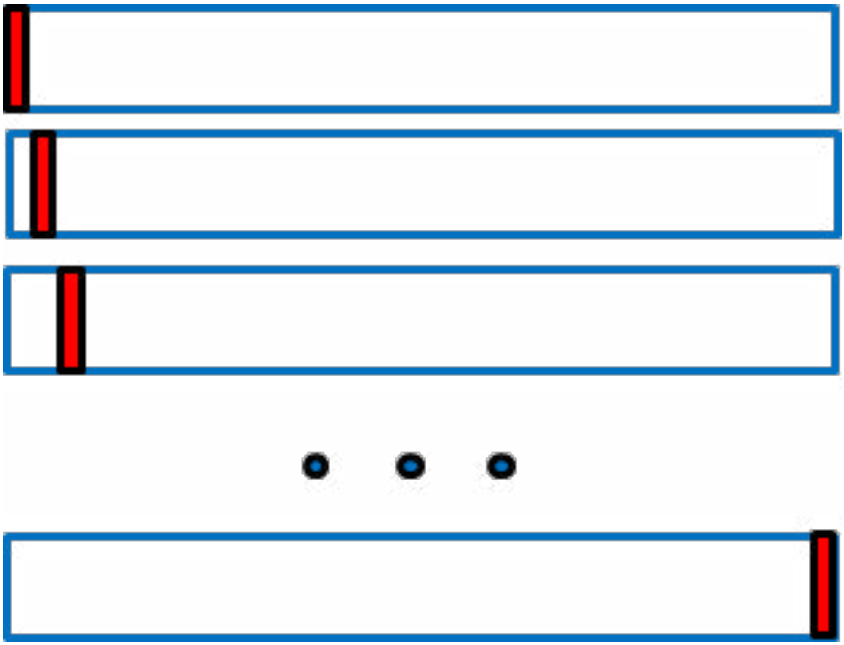
Zasada działania podobna do **walidacji *n*-krotnej**, przy czym podzbiory tworzy się przy wykorzystaniu losowania ze zwracaniem. Dlatego podzbiory wykorzystane do walidacji mogą zawierać powtarzające się elementy. Zaletą tej metody jest możliwość kształtowania dowolnych proporcji pomiędzy liczebnościami zbioru uczącego i testowego (w przypadku walidacji *n*-krotnej proporcja ta zależy od liczby *n*). Pewne ograniczenie tej metody wynika z faktu, że nie wszystkie elementy zbioru wejściowego muszą brać udział w **walidacji krzyżowej** (w wyniku losowania nie ma gwarancji, że na pewno znajdą się w którymś z podzbiorów walidacyjnych).



Na rysunku pokazano przykładowy podział zbioru **przypadków uczących** na trzy zbiory walidacyjne: rozłączne dla metody walidacji 3-krotnej oraz częściowo się pokrywające dla metody *bootstrap*. W tym drugim przypadku (założono tworzenie 3 podzbiorów walidacyjnych), zawierają one wspólne elementy. Dla obydwu metod do podzbioru uczącego należą wszystkie pozostałe elementy zbioru danych wejściowych.

Walidacja metodą *leave-one-out*

Jest to metoda walidacji stosowana w przypadku, gdy liczebność zbioru **danych uczących** sieci jest niewielka. Wtedy zbiór ten jest dzielony na jednoelementowe podzbiory. Każdy taki pojedynczy element służy do testowania sieci, jej uczenie następuje z wykorzystaniem wszystkich pozostałych elementów. Czynność ta jest powtarzana tyle razy, ile jest elementów w zbiorze danych wejściowych. W istocie walidacja metodą *leave-one-out* jest walidacją *N*-krotną, gdzie *N* jest liczbą przypadków uczących.



Błąd sieci określa się jako sumę przypadków jej błędnego działania (błędnej klasyfikacji lub predykcji) uzyskanych dla pojedynczych danych wejściowych.

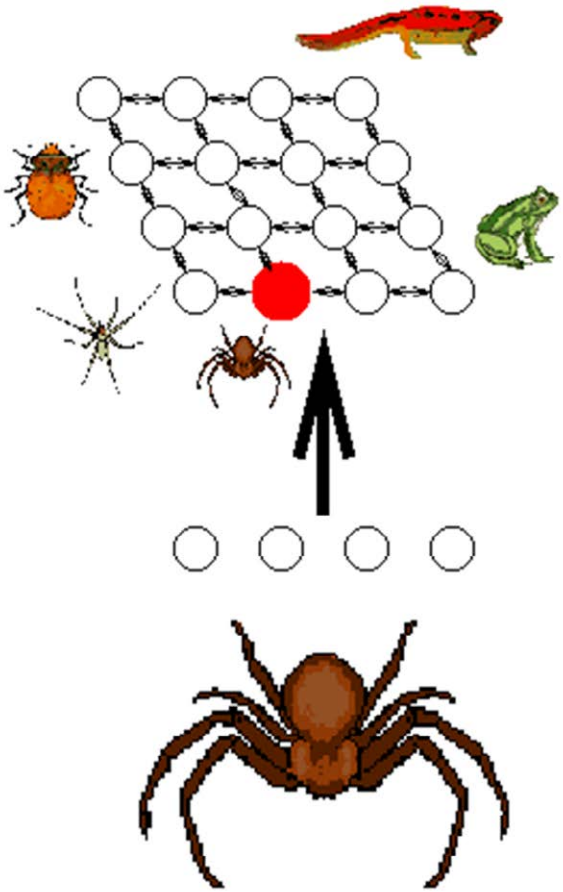
Walidacja *n*-krotna

Przy tym sposobie walidacji zbiór **danych uczących** jest dzielony na *n* rozłącznych podzbiorów (zwykle równolicznych). Jeden z nich służy do testowania sieci, a pozostałe do jej uczenia. Operacja ta powtarzana jest *n* razy, tak, aby każdy podzbiór pełnił rolę **zbioru walidacyjnego**. Skuteczność sieci określa się zwykle jako średnią błędów klasyfikacji lub predykcji uzyskanych dla poszczególnych zbiorów walidacyjnych.

Walidacja krzyżowa k-krotna (dla k=4)

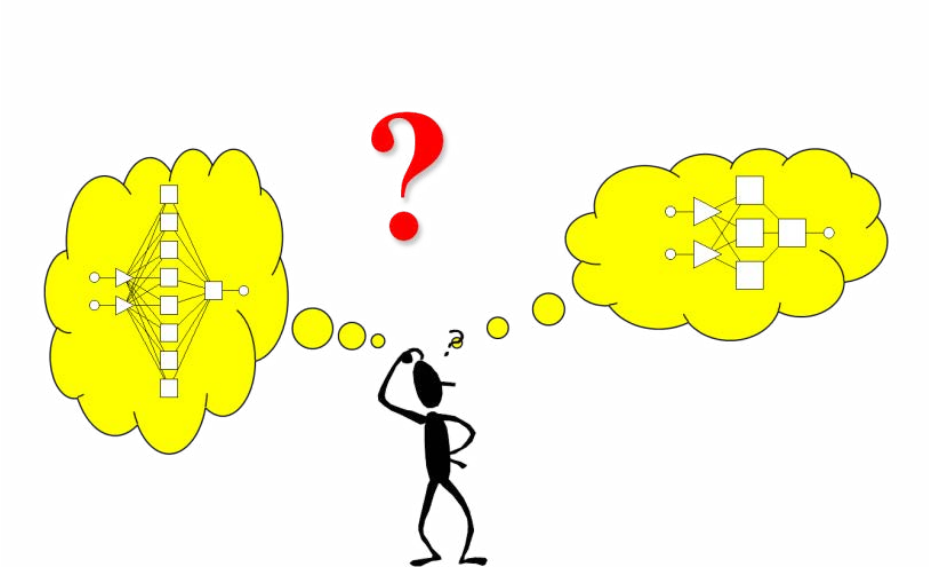
Warstwa topologiczna

Warstwa w **sieci Kohonena**, na której prezentowany jest wynik pracy tej sieci w postaci **mapy topologicznej**. Po procesie uczenia neurony warstwy topologicznej mają przypisane do siebie obiekty (zestawy **danych wejściowych**), które pojawiały się szczególnie często podczas procesu **samouczenia sieci**. Neurony warstwy topologicznej mogą więc te obiekty wykrywać, sygnalizować, grupować i kategoryzować. Przykład warstwy topologicznej przedstawiono na rysunku.



Warstwa ukryta

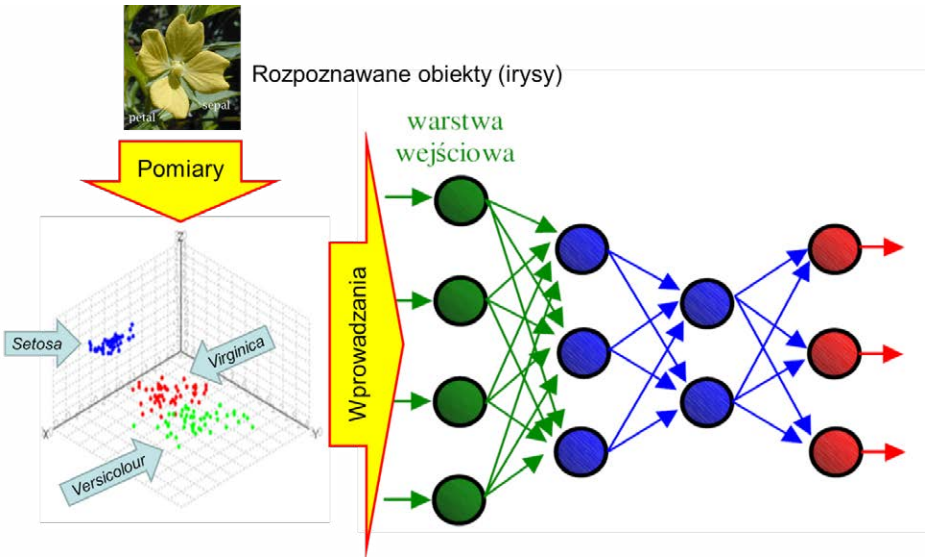
Warstwę określa się jako ukrytą, jeśli do **neuronów** tej warstwy nie ma bezpośredniego dostępu ani od strony **wejścia** sieci, ani od strony jej **wyjścia**. Rola warstwy ukrytej polega na przetwarzaniu **danych wejściowych** w taki sposób, żeby uzyskane **dane wyjściowe** były przydatne dla potrzeb wypracowania rozwiązania całego zadania w **warstwie wyjściowej** sieci. Twórca sieci neuronowej ma zwykle trudności z ustaleniem, ile elementów powinna mieć warstwa ukryta.



Mała liczba neuronów w warstwie ukrytej może sprawić, że sieć będzie zbyt prymitywna, żeby poradzić sobie z trudnością rozwiązywanego zadania. Duża liczba neuronów w warstwie ukrytej da sieć znacznie inteligentniejszą, ale spowoduje, że sieć sprawi więcej kłopotów podczas **uczenia**. Liczbę neuronów w warstwie ukrytej optymalizuje się dla danego problemu metodą empiryczną, analizując jakość działania sieci o różnej ilość neuronów ukrytych. Odrębnym problemem jest też **liczba warstw ukrytych**.

Warstwa wejściowa

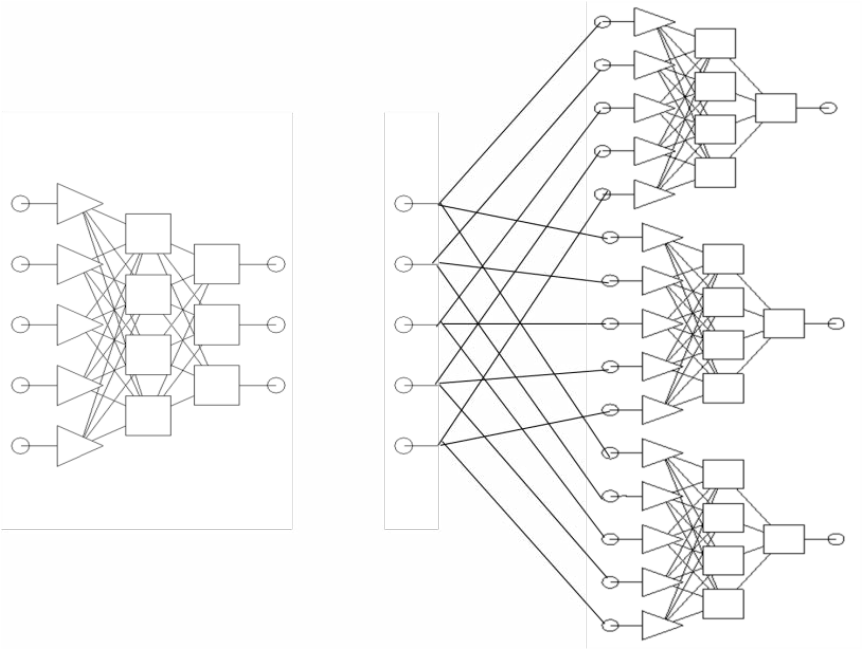
Jest to warstwa **sieci neuronowej**, która nie uczestniczy bezpośrednio w wypracowywaniu **odpowiedzi sieci**. Rola **neuronów** warstwy wejściowej polega na tym, że rozprowadzają one **dane wejściowe** w postaci **sygnałów** do neuronów pierwszej warstwy ukrytej. Korzystną okolicznością związaną z warstwą wejściową jest to, że także w tej warstwie może w wyniku **procesu uczenia** dochodzić do **redukcji połączeń**. Jeśli redukcja taka wskaże na celowość usunięcia niektórych **neuronów** warstwy wejściowej – będzie to dowodziło, że odpowiednie **dane wejściowe** były niepotrzebne. Ma to spore znaczenie praktyczne, bo twórca **sieci neuronowej** często nie wie, które z posiadanych **danych wejściowych** są przydatne do rozwiązania postawionego zadania, a które nie. Jeśli tylko **zbiór uczący** jest wystarczająco liczny, można sobie pozwolić na eksperyment polegający na tym, że na wejście sieci podamy wszystkie posiadane dane i uruchomimy **proces uczenia**. Jeśli w wyniku uczenia otrzymamy sieć charakteryzującą się na **egzaminie** małą wartością **błędu**, a jednocześnie w sieci dojdzie do znaczącej redukcji połączeń związanych z warstwą wejściową, to można uznać, że w wyniku **uczenia** sieć wybrała użyteczne **dane wejściowe**, a odrzuciła dane nieprzydatne. Hasło **redukcja warstwy wejściowej** pokazuje na konkretnym przykładzie (do którego nawiązuje rysunek poniżej), jak to działa.



Warstwa wyjściowa

Zbiór neuronów, których **sygnały wyjściowe** są traktowane jako **wyjście** całej **sieci**, określany jest mianem warstwy wyjściowej **sieci neuronowej**. Dla twórcy sieci neuronowej niebanalny bywa problem, ile **neuronów** w **warstwie wyjściowej** powinna mieć budowana sieć. Pozornie odpowiedź jest łatwa: Ze sposobu sformułowania zadania wynika, ile **danych wyjściowych** oczekuje użytkownik sieci, a z tego z kolei wynika, ile neuronów powinna liczyć warstwa wyjściowa sieci.

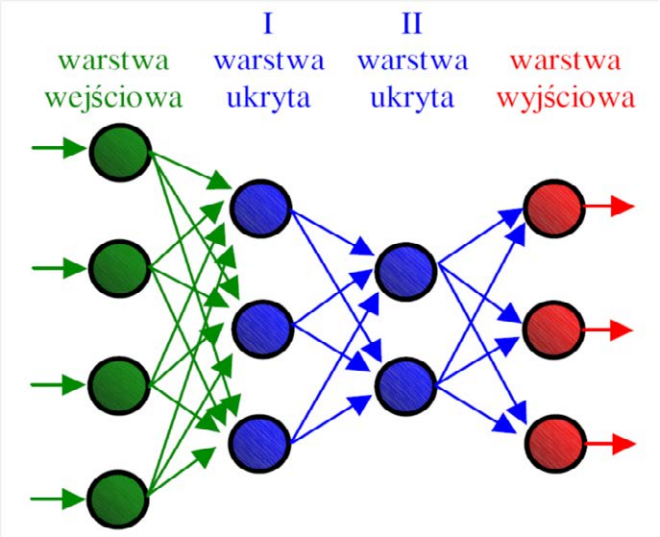
Ta odpowiedź jest jednak niepełna, ponieważ w wielu przypadkach kwestią całkowicie otwartą jest to, czy potrzebną liczbę **danych wyjściowych** lepiej jest uzyskać poprzez zastosowanie jednej sieci o wielu **wyjściach**, czy też zastosować kilka sieci, z których każda dostarcza jednej danej wyjściowej.



Obie rozważane wyżej możliwości mają różne konsekwencje w odniesieniu do **uczenia** sieci. Zagadnienie to naświetlone jest dodatkowo w haśle **wyjście**.

Warstwy w sieciach neuronowych

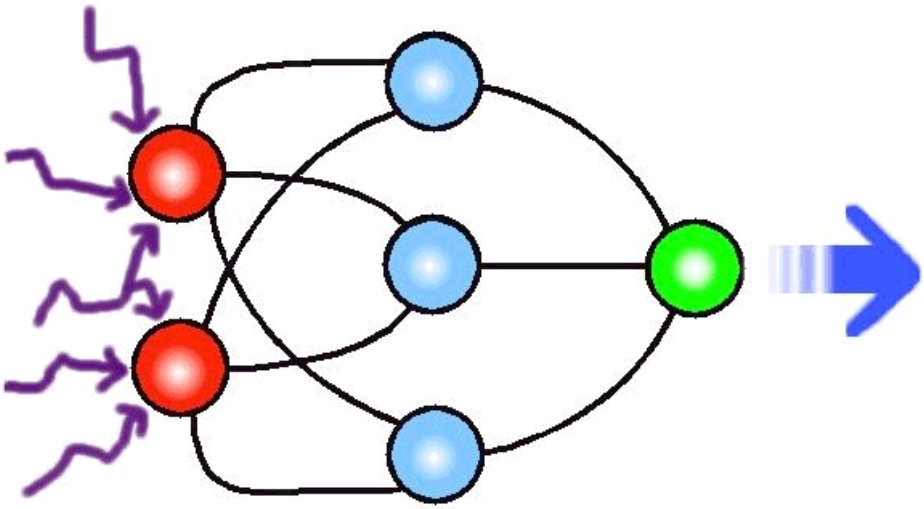
Struktura sieci neuronowej zakłada zwykle istnienie warstw **neuronów**. Warstwa stanowi umowne wyobrażenie sposobu ułożenia grupy neuronów mających w sieci określoną funkcję. Wyróżnia się **warstwę wejściową**, **warstwy ukryte** oraz **warstwę wyjściową**.



Podział sieci na warstwy determinuje między innymi **połączenia neuronów**. Najczęściej łączone są tylko neurony należące do sąsiednich warstw.

Wejście

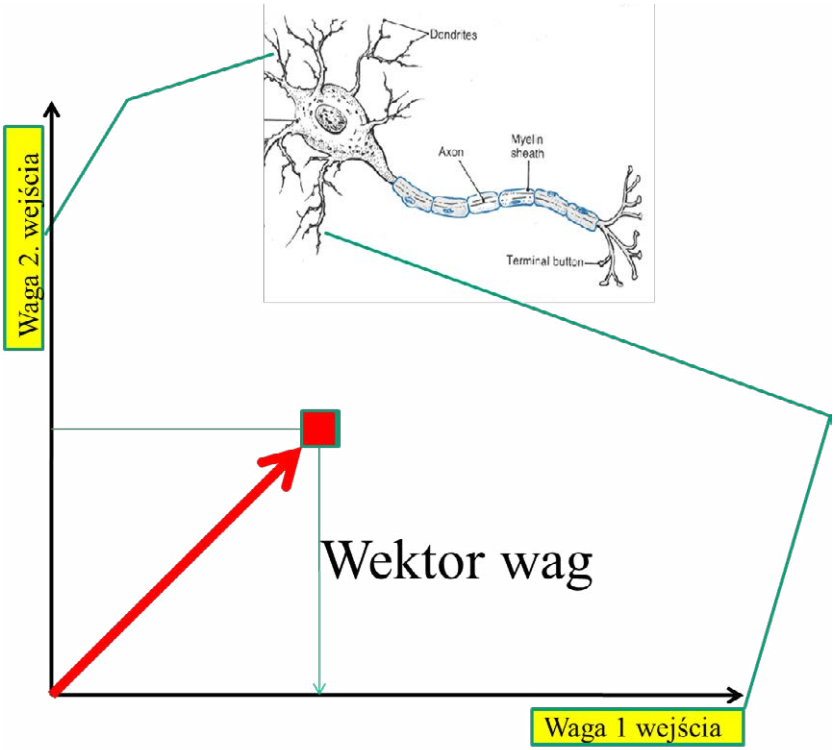
Skrótowa nazwa drogi wprowadzania do poszczególnych neuronów albo do całej sieci neuronowej **danych wejściowych**. Czasami żargonowo nazwą tą określa się same **dane wejściowe**, chociaż jest to niepoprawne.



Zaletą sieci neuronowej jest to, że można na jej **wejście** wprowadzać niezrozumiałe dla użytkownika i nieuporządkowane **dane wejściowe**, a po nauczeniu sieci na jej **wyjściu** dostaje się zwykle porządny i użyteczny wynik, wygodny i łatwy do interpretacji.

Wektor wag

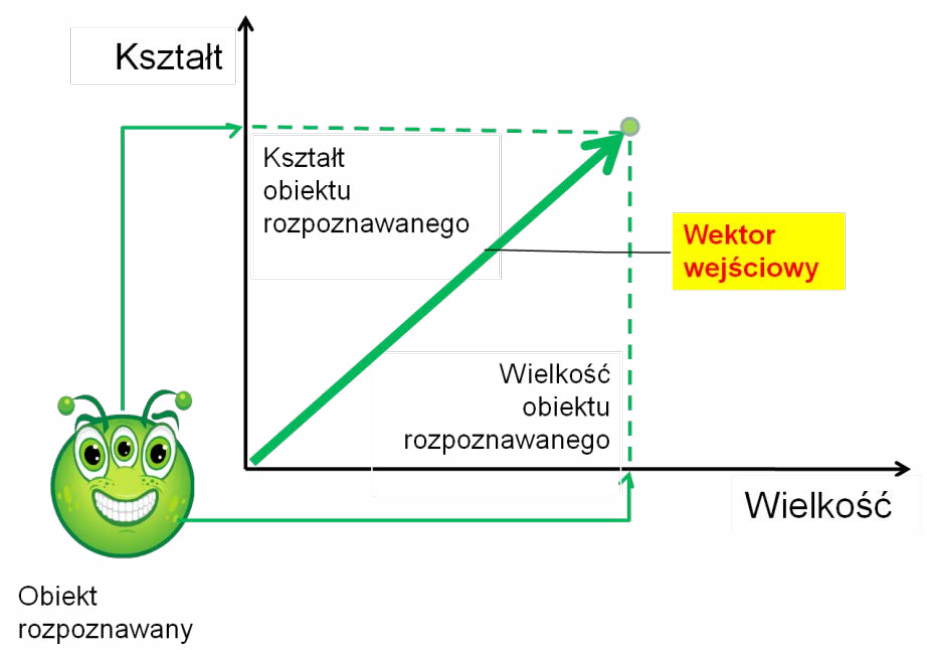
Parametry neuronu, jakimi są **wagi**, są uporządkowane i ponumerowane, podobnie jak poszczególne **dane wejściowe**, z którymi te wagi są skojarzone. Dlatego wygodnie jest czasem mówić o całym zestawie **wag** danego **neuronu** jako o wektorze wag. Wektor wag determinuje właściwości neuronu, bo każda jego zmiana powoduje, że neuron zmienia swoje zachowanie, czasem bardzo radykalnie (patrz rysunek przy haśle **wagi**). Na zasadzie analogii do wektora wag również zestaw **danych wejściowych** (składowych **sygnału wejściowego**) określany bywa jako **wektor wejściowy** albo krótko jako **sygnał**. Wzajemne położenie wektora wag (będącego wynikiem **procesu uczenia**) oraz **wektora wejściowego** decyduje o **odpowiedzi neuronu**. Wynika to z przyjmowanej najczęściej **agregacji danych wejściowych** w postaci agregacji liniowej.



Na zasadzie analogii z wektorem wag pojedynczego neuronu mówi się także o wektorze wag całej sieci – rozumiejąc przez to konkatenację (połączenie) wektorów wag wszystkich neuronów wchodzących w skład sieci.

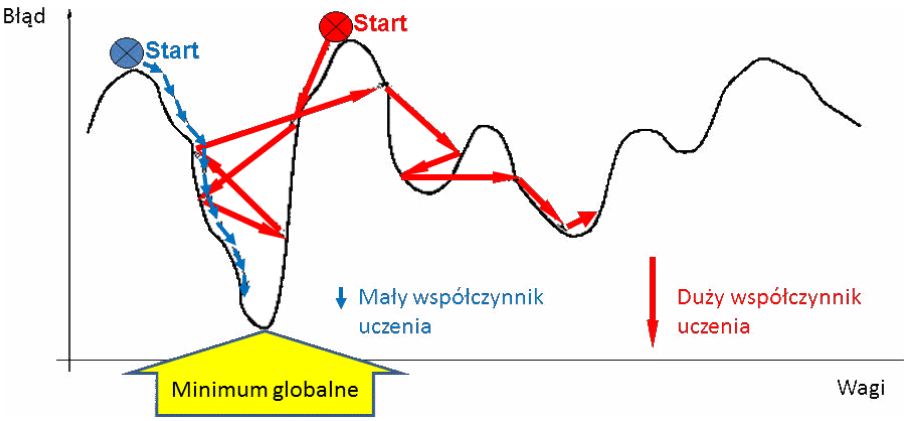
Wektor wejściowy

Zestaw **danych wejściowych** podawanych na **wejście neuronu** jest ponumerowany. Numeracja ta jest arbitralnie wprowadzona dla wygody operowania danymi wejściowymi oraz dla uproszczenia opisujących je formuł matematycznych. Jednak korzystając z faktu tego ponumerowania **wejść** mówi się często o **danych wejściowych** podawanych do pojedynczego **neuronu** albo do całej **sieci** jako o wektorze wejściowym. Składowe tego wektora opisują obiekt, który jest przedmiotem rozpoznawania w **sieciach klasyfikacyjnych** albo dla którego należy obliczyć wybrane **wartości wyjściowe** w **sieciach regresyjnych**.



Współczynnik uczenia

Współczynnik uczenia (*learning rate*) jest parametrem wiążącym *lokalne* właściwości **funkcji błędu sieci neuronowej**, wyznaczane na przykład z pomocą algorytmu **wstecznej propagacji błędu**, odwołujące się (w procesie różniczkowania) do nieskończenie małych zmian wag – z działaniem polegającym na makroskopowych (a więc nie nieskończenie małych) zmianach wag w każdym kolejnym kroku uczenia. **Algorytm uczenia** wskazuje, w jakim kierunku należy zmienić wagi, żeby błąd popełniany przez sieć zmalał, natomiast wybór **współczynnika uczenia** decyduje o tym, jak bardzo zdecydujemy się te wagi we wskazanym kierunku zmienić. Jeśli współczynnik uczenia wybierzemy zbyt mały, to proces uczenia może bardzo długo trwać, bo będziemy bardzo wolno zmierzać do finalnego (optymalnego) zestawu wartości wszystkich wag. Jeśli jednak zastosujemy zbyt duży współczynnik uczenia – to będziemy wykonywać zbyt duże kroki i na skutek niemonotonicznej charakterystyki funkcji błędu może się zdarzyć, że „przeskoczymy" właściwą drogę zmierzającą do punktu zapewniającego minimum funkcji błędu. W efekcie błąd po wykonaniu poprawki wag może być większy, a nie mniejszy niż poprzednio (patrz rysunek).



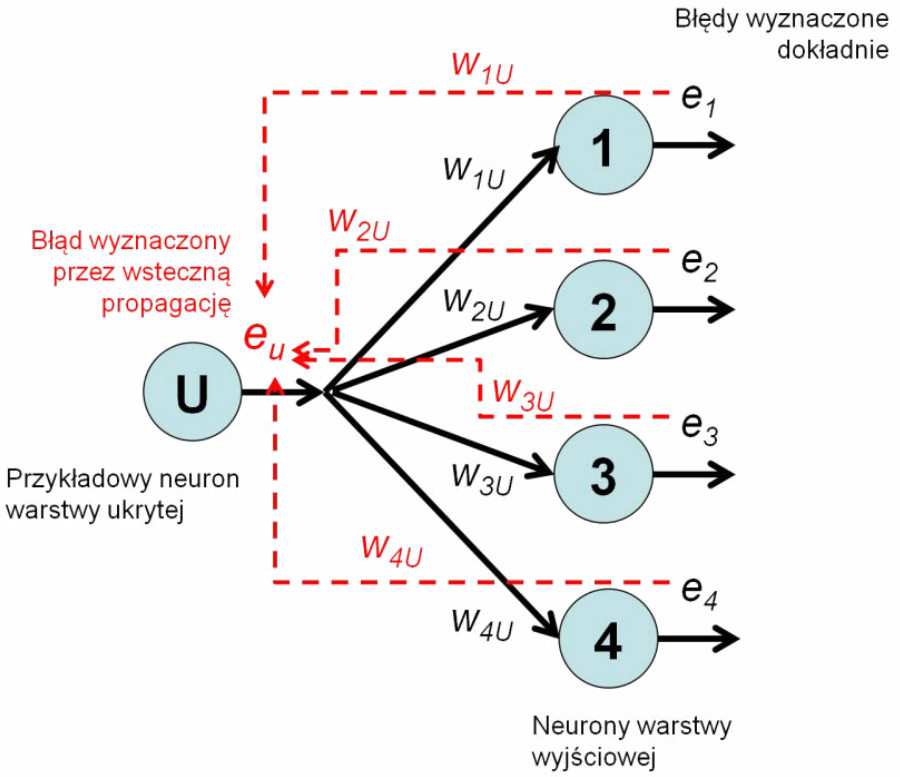
W skrajnym przypadku zbyt duża wartość współczynnika uczenia może prowadzić do niestabilności procesu uczenia, gdyż wartości wag zamiast się stabilizować w trakcie uczenia – uciekają do nieskończoności.

Użyteczną metaforą przybliżającą intuicyjne zrozumienie znaczenia współczynnika uczenia jest porównanie go do **surowości nauczyciela**.

Wsteczna propagacja błędów

Zasada ustalania wartości **błędów** dla **neuronów** należących do **warstw ukrytych** wykorzystywana przez **algorytm uczenia** sieci neuronowej. Wsteczna propagacja błędów jest wykorzystywana przez algorytm *backpropagation*, historycznie pierwszy i do dziś jeden z najpopularniejszych algorytmów uczenia z nauczycielem. Algorytm ten bywa stosowany do uczenia różnego typu **sieci jednokierunkowych**. Opiera się na koncepcji poprawiania na każdym kroku **procesu uczenia** wartości korekty wag na podstawie oceny **błędu** popełnianego przez każdy neuron podczas **uczenia** sieci.

Konieczność stosowania wstecznej propagacji błędu wynika z tego, że tylko błędy w neuronach warstwy wyjściowej wyznacza się bezpośrednio na podstawie **danych wyjściowych** i **odpowiedzi wzorcowych** zawartych w **zbiorze uczącym**. Natomiast dla neuronów w **warstwach ukrytych** błąd musi być wyznaczany właśnie poprzez **wsteczną propagację**. Przy tej wstecznej propagacji rozważany neuron otrzymuje (ma przypisaną) wartość **błędu** wyliczaną na podstawie wartości **błędów** wszystkich tych neuronów, do których wysyłał on wartość swojego sygnału wyjściowego jako składnika ich **danych wejściowych**. Przy obliczaniu wartości wstecznie rzutowanego błędu uwzględnia się wartości **wag** połączeń pomiędzy rozważanym neuronem i neuronami, których błędy są do niego wstecznie rzutowane (wstecznie, bo przeciwnie do kierunku przepływu sygnału w **jednokierunkowej sieci**).

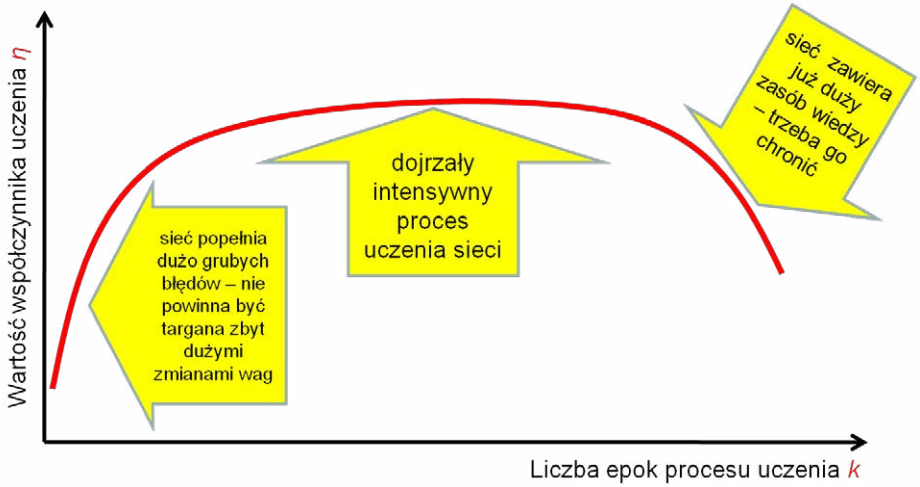


Po obliczeniu wartości błędów neuronów warstwy ukrytej najbliższej **wyjścia** procedurę się powtarza, przyjmując wyliczone wartości błędów jako znane i w taki sam sposób przeprowadzając ich propagację do kolejnej warstwy ukrytej, bliższej **wejścia**.

Wybór współczynnika uczenia

Współczynnik uczenia decyduje o szybkości procesu uczenia sieci neuronowej i powinien być dostosowany do charakteru **funkcji błędu**. Ponieważ jednak przy typowych zastosowaniach sieci neuronowych charakter funkcji błędu jest nieznany – dlatego wybór wartości współczynnika uczenia jest z reguły całkowicie arbitralny, a jego ocena dokonywana jest *ex post*, na podstawie obserwacji przebiegu procesu uczenia. Niektóre **programy modelujące sieci neuronowe** proponują użytkownikowi sugerowaną wartość współczynnika uczenia (wynikającą z doświadczeń twórców oprogramowania), a użytkownik wartość tę po prostu zatwierdza (lub zmienia, jeśli ma powody). Niektóre **algorytmy uczenia** sieci neuronowych (na przykład **Quickpropagation**) zawierają w sobie mechanizm automatycznej optymalizacji wartości współczynnika uczenia, co powoduje, że początkowy wybór wartości tego współczynnika wykonywany przez użytkownika ma charakter jedynie „wstępnej przymiarki".

Godna uwagi strategia zmian współczynnika uczenia w trakcie uczenia sieci przedstawiona jest na rysunku poniżej.

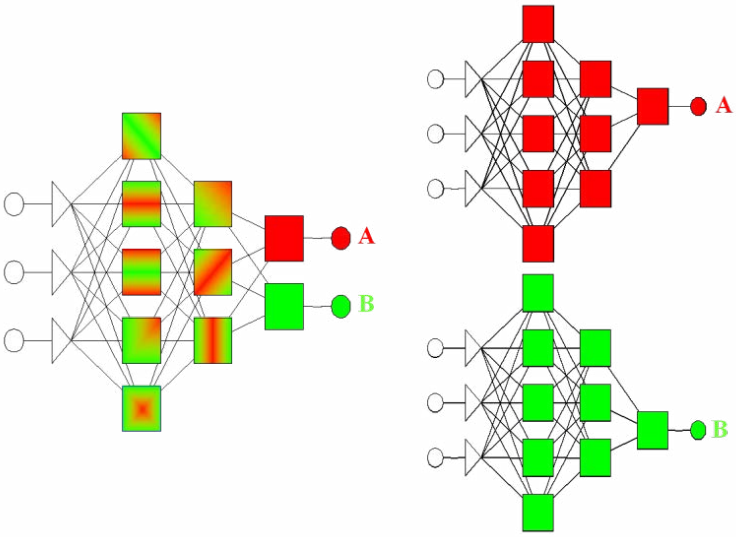


Wyjście

Skrótowa nazwa drogi wyprowadzania z poszczególnych neuronów albo z całej sieci neuronowej **danych wyjściowych**. Czasami żargonowo nazwą tą określa się same **dane wyjściowe**, chociaż jest to niepoprawne.

Każdy **neuron** posiada swoje **wyjście**, na którym wytwarza **sygnał** wyjściowy wynikający z zastosowanej metody **agregacji danych wejściowych** oraz z przyjętej **funkcji aktywacji**. Jednak pojęcie wyjścia pojedynczego **neuronu** jest dokładnie omówione w kontekście hasła **neuron** i nie ma potrzeby tu się nim osobno zajmować.

Natomiast niebanalne problemy wynikają przy dyskusji wyjścia z całej **sieci**. Z zasady przyjmuje się, że wyjściem całej sieci jest zbiór wartości pojawiających się na wyjściach neuronów budujących **warstwę wyjściową** sieci. Jednak bywają okoliczności, w których warto od tej zasady odstąpić. Załóżmy, że uczymy jedną sieć neuronową o dwóch wyjściach A oraz B (patrz rysunek po lewej stronie). W neuronach obu warstw ukrytych będą musiały być zgromadzone informacje potrzebne do wyznaczania wartości A oraz B. Czasem może to być korzystne, gdy między wyjściami zachodzi synergia i, doskonaląc pracę sieci zmierzającą do wyznaczania poprawnych wartości A, przy okazji gromadzi się wiedzę przydatną do wyznaczania wartości B.

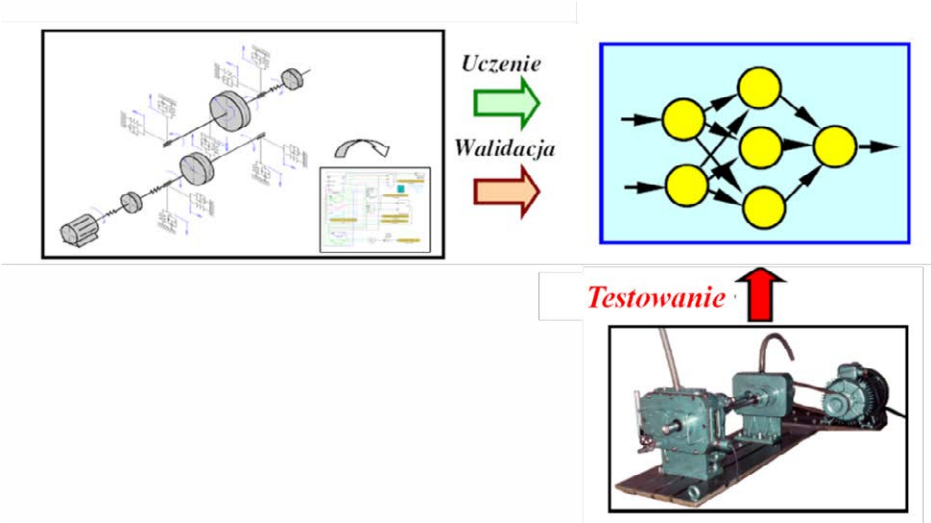


Częściej jednak bywa tak, że między wyjściami jest konflikt i wyznaczając wartości przydatne do obliczenia A psujemy wartości potrzebne dla B – i *vice versa*. Lepiej jest wtedy zbudować dwie osobne sieci (patrz rysunek po prawej stronie). Każdą z nich można wtedy będzie w całości optymalnie dostroić do obliczania wymaganego wyjścia – odpowiednio A albo B.

Zbiór testowy

Część **zbioru uczącego** przeznaczona do przeprowadzenia po zakończeniu **uczenia** jednorazowej kontroli (czy w wyniku zbiegu okoliczności mimo okresowej walidacji nie doszło w trakcie **uczenia** do utraty zdolności **generalizacji**). Zbiór testowy tworzy się wyłącznie w przypadku posiadania bardzo dużego **zbioru uczącego**, włączając do niego około 10% losowo wybranych **przypadków uczących**.

Wyjątkowo ciekawy przykład użycia zbioru testowego przedstawia rysunek, zaczerpnięty (z drobnymi przeróbkami) z pracy doktorskiej prof. Piotra Czecha z Politechniki Śląskiej. We wspomnianej pracy doktorskiej chodziło o wykrywanie uszkodzeń przekładni zębatych. Otóż **uczenie** i **walidacja sieci** prowadzone były na podstawie danych uzyskanych z komputerowej symulacji rozważanej przekładni, natomiast do testowania **sieci** użyto danych pochodzących z rzeczywistej przekładni, w której celowo wprowadzano uszkodzenia i sprawdzano, czy sieć je wykryje. Jest to wzorcowy wręcz przykład zbioru testowego.



Na koniec jeszcze jedna uwaga: Wielkość **błędu** ustalona przy użyciu zbioru testowego powinna być zbliżona do wielkości błędu ustalanego dla **zbioru walidacyjnego**. Jeśli błąd uzyskany dla zbioru testowego jest wyraźnie większy od błędu dla **zbioru walidacyjnego**, to uczenie sieci trzeba powtórzyć, ponownie dzieląc w sposób losowy **zbiór uczący** i wydzielając nowy **zbiór walidacyjny** oraz **zbiór testowy**.

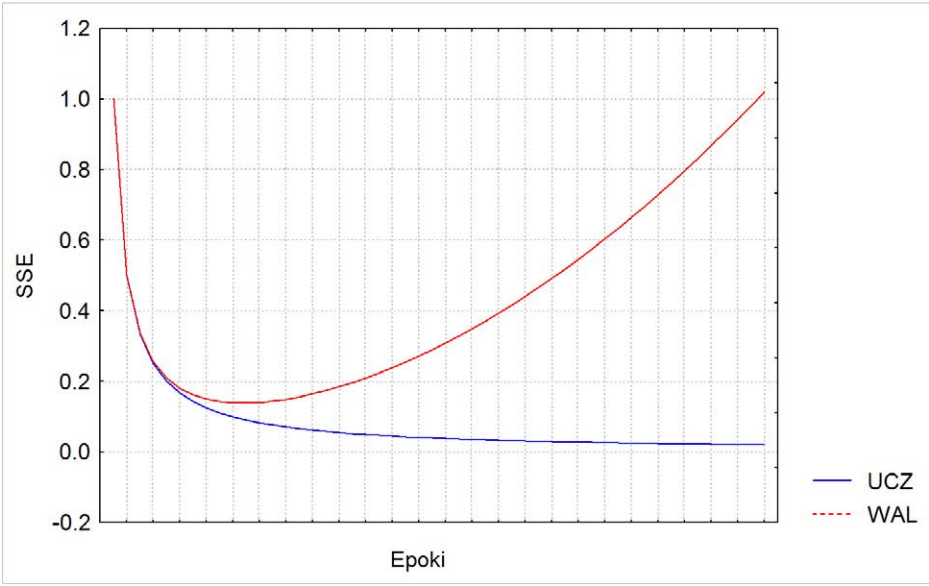
Zbiór uczący

Zbiór **przypadków uczących**, czyli zadań zawierających **dane wejściowe** oraz skojarzone z nimi **odpowiedzi wzorcowe**. Część zbioru uczącego wykorzystywana jest do uczenia sieci z użyciem wybranego **algorytmu uczenia**, natomiast część (zwykle około 20%) zostaje losowo wydzielona jako **zbiór walidacyjny**, wykorzystywany w procesie uczenia jedynie pośrednio. Jeśli zbiór uczący jest bardzo liczny, to czasem można na jego bazie wydzielić dodatkowo (losowo!) dodatkowy **zbiór testowy** (około 10% losowo wybranych **przypadków uczących**).

Na rysunku pokazano przykładowy zbiór uczący wykorzystywany przy uczeniu sieci rozpoznawania twarzy ludzkich. Do każdej z fotografii dołączona jest oczywiście informacja na temat poprawnej identyfikacji osoby z portretu, która wykorzystywana jest podczas uczenia do oceny rozpoznań podawanych przez sieć.

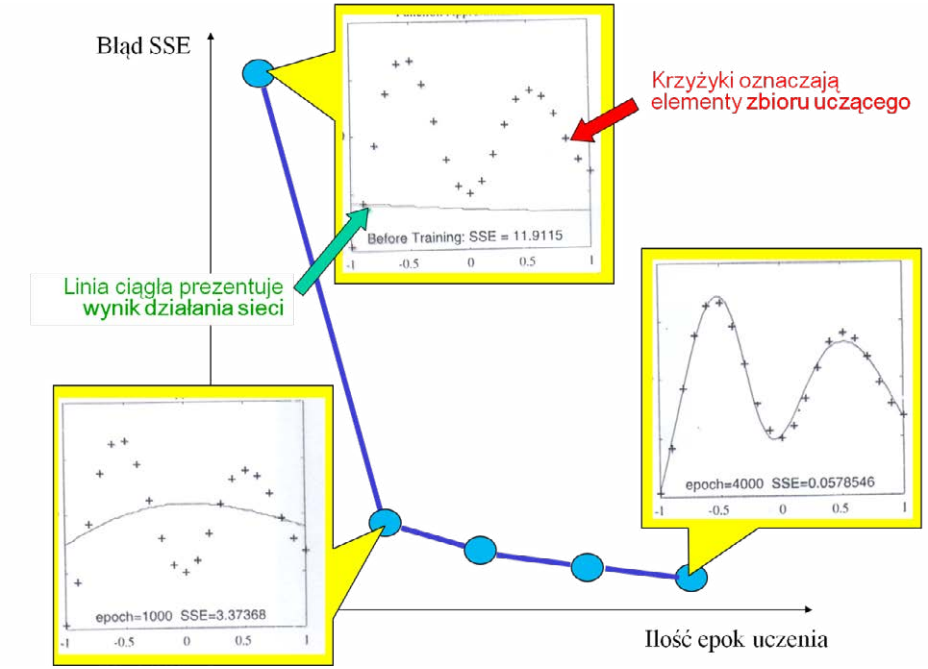
Zbiór walidacyjny

Część **zbioru uczącego** (zwykle losowo wybranych około 20% **przypadków uczących**) przeznaczona do przeprowadzania w trakcie **uczenia** okresowej **walidacji** mającej na celu zapobieganie zjawisku **przeuczenia**. Obserwując zmiany **błędu SSE** w trakcie uczenia sieci można zaobserwować dwa zjawiska: systematyczne malenie błędu dla **zbioru uczącego** (bo **algorytm uczenia** jest ukierunkowany na minimalizację tego błędu) oraz początkowo malejący błąd dla zbioru walidacyjnego (gdy sieć zachowuje zdolność do **generalizacji**), a potem błąd rosnący dla tego zbioru (co wiąże się ze zjawiskiem **przeuczenia**). Moment zapoczątkowania wzrostu błędu dla zbioru walidacyjnego stanowi sygnał do zakończenia szkolenia sieci i odzyskania najlepszych jej wag z epoki poprzedzającej początek wzrostu tego błędu.



Zmiany błędu

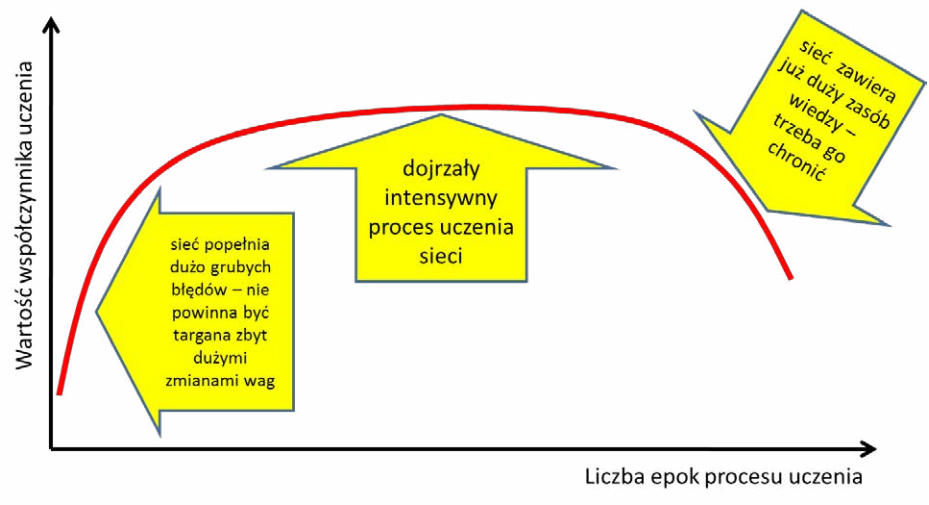
Zmiany błędu obserwowane w trakcie **procesu uczenia** charakteryzują się zwykle tym, że na początku przebiegają bardzo dynamicznie (błąd maleje bardzo szybko, a wartość jego zmniejszenia jest początkowo bardzo duża), a potem maleje znacznie wolniej i spadek błędu jest niewielki. Zdawać by się mogło, że najważniejszy jest ten początkowy okres uczenia sieci (gdy błąd szybko maleje), natomiast doskonalenie działania sieci w obszarze powolnego malenia niewielkich już błędów ma pozornie niewielkie znaczenie.



Ocena ta nie jest jednak trafna. Rysunek pokazuje, że początkowy szybki okres malenia błędu w istocie związany jest z tym, że sieć neuronowa dostosowuje wtedy jedynie zgrubnie swoje działanie do właściwości rozwiązywanego zadania. Natomiast ten pozornie niewielki spadek błędu w powolnym okresie uczenia prowadzi do istotnego dopasowania działania sieci do wzorców zawartych w zbiorze uczącym.

Zmiany wartości współczynnika uczenia

Współczynnik uczenia może zmieniać się w trakcie **uczenia** sieci, co zwykle polepsza jakość uzyskiwanych wyników. Optymalne wydają się niemonotoniczne zmiany wartości współczynnika uczenia w trakcie nauki, pokazane i uzasadnione na rysunku poniżej.



Bibibliografia

1. Tadeusiewicz R., *Sieci neuronowe*, Akademicka Oficyna Wydawnicza, Warszawa 1993 (wyd. I i II tego samego roku). http://winntbg.bg.agh.edu.pl/skrypty/0001/
2. Korbicz J., Obuchowicz A., Uciński D., *Sztuczne sieci neuronowe: podstawy i zastosowania*, Akademicka Oficyna Wydawnicza, Warszawa, 1994.
3. Herz J., Krogh A., Palmer R.G., *Wstęp do teorii obliczeń neuronowych*, Wydawnictwa Naukowo-Techniczne, Warszawa 1995.
4. Jędruch W., *Sztuczne sieci neuronowe*, Wydawnictwo Naukowe PWN, Warszawa 1996.
5. Żurada J., Barski M., Jędruch W., *Sztuczne sieci neuronowe. Podstawy teorii i zastosowania*, Wydawnictwo Naukowe PWN, Warszawa1996.
6. Osowski S., *Sieci neuronowe w ujęciu algorytmicznym*, Wydawnictwo Naukowo-Techniczne, Warszawa 1996.
7. Master T., *Sieci neuronowe w praktyce*, Wydawnictwa Naukowo-Techniczne, Warszawa 1996.
8. Rutkowska D., Piliński M., Rutkowski L., *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*, Wydawnictwo Naukowe PWN, Warszawa 1997.
9. *Sieci Neuronowe*, t. 6 monografii *Biocybernetyka i Inżynieria Biomedyczna 2000*, red. W. Duch, J. Korbicz, L. Rutkowski, R. Tadeusiewicz, Polska Akademia Nauk, Wydawnictwo EXIT, Warszawa 2000.
10. *Sieci neuronowe w inżynierii biomedycznej*, t. 9 monografii: *Inżynieria biomedyczna. Podstawy i zastosowania*, red. R. Tadeusiewicz, J. Korbicz, L. Rutkowski, W. Duch, Polska Akademia Nauk, Wydawnictwo EXIT, Warszawa 2013.