

RYSZARD TADEUSIEWICZ  
TOMASZ GĄCIARZ, BARBARA BOROWIK, BARTOSZ LEPEK

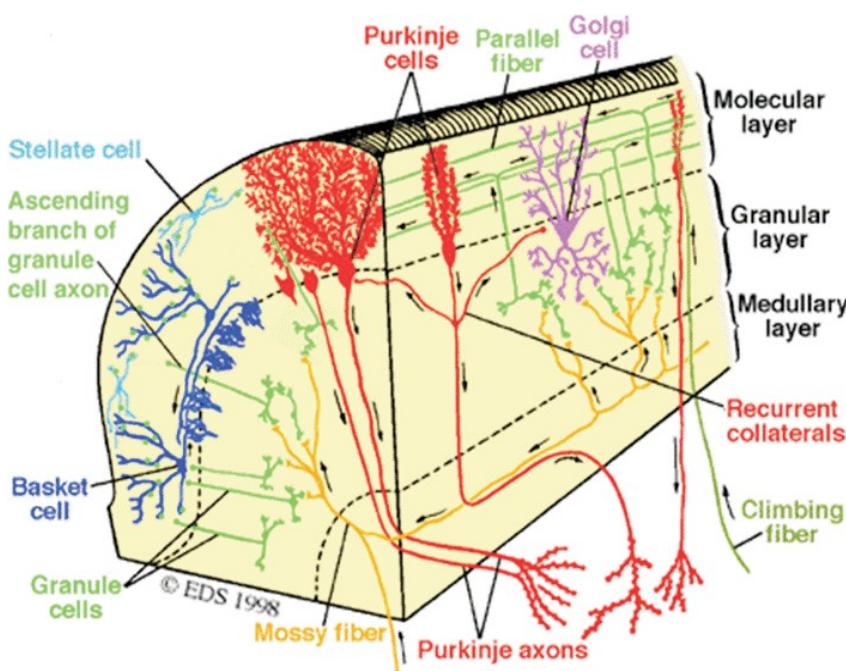
## ODKRYWANIE WŁAŚCIWOŚCI SIECI NEURONOWYCH

PRZY UŻYCIU PROGRAMÓW W JĘZYKU C#

## 2. Struktura sieci

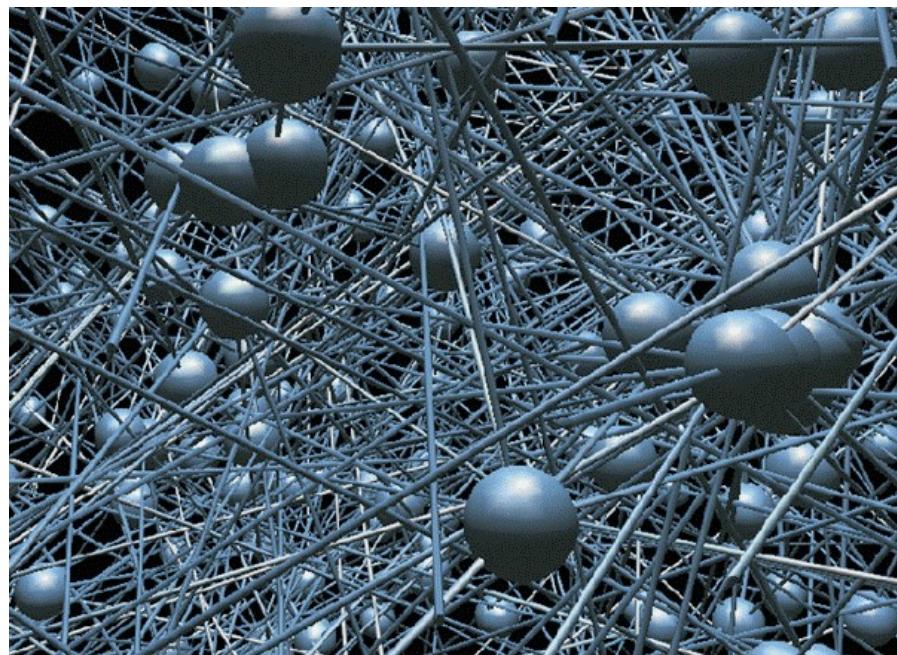
### 2.1. Jak to jest zbudowane?

Któż z nas nie zaczynał poznawania świata od rozkręcenia budzika lub rozłupania magnetofonu – żeby zobaczyć, co jest w środku? Dlatego zanim opowiem Ci o tym, jak sieć działa i jak jej używać – postaram się w miarę dokładnie i w miarę prosto opisać, jak jest ona zbudowana. Jak już wiesz z poprzedniego rozdziału – sieć neuronowa jest systemem dokonującym określonych obliczeń na zasadzie równoczesnej pracy wielu połączonych ze sobą elementów zwanych neuronami. Taka budowa zaobserwowana została pierwotnie w biologicznym systemie nerwowym (na przykład w mózdku człowieka, którego fragment schematycznie przedstawiłem na rysunku 2.1).



Rys. 2.1. Schematyczny obraz kory mózdku pokazuje, że biologiczny system nerwowy zbudowany jest z wielu połączonych ze sobą neuronów. Ta sama zasada obowiązuje w odniesieniu do sztucznych sieci neuronowych

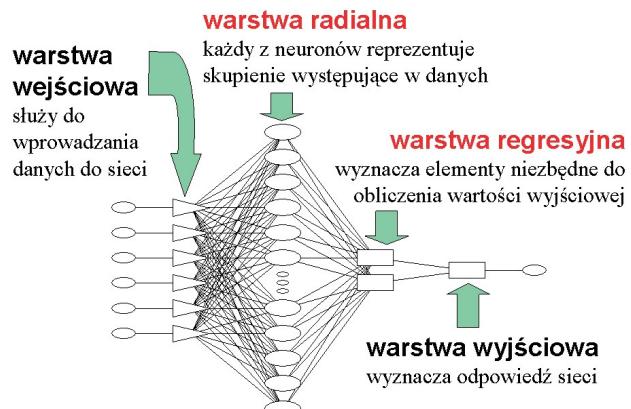
Sieci neuronowe są także zbudowane z wielu neuronów, tyle tylko że sztucznych, to znaczy znacznie uproszczonych w stosunku do oryginału, a także znacznie prościej (by nie powiedzieć – prymitywniej) ze sobą połączonych. Gdyby spróbować zbudować sztuczną sieć neuronową wzorując się na strukturze rzeczywistego systemu nerwowego, to powstały twór byłby bardzo mało przejrzysty i w dodatku trudno by się go kontrolowało. Na rysunku 2.2 pokazałem w przybliżeniu, jak wyglądałaby sieć sztucznych neuronów zbudowana według tych samych schematów strukturalnych, jakie można odnaleźć w rzeczywistym systemie nerwowym. Łatwo zauważysz, że taka struktura nie jest przyjazna do prowadzenia z jej użyciem jakichkolwiek eksperymentów. Przeciwnie – można się w niej zgubić jak w lesie!



Rys. 2.2. Sztuczna sieć o strukturze wzorowanej na trójwymiarowej mapie mózgu byłaby niewygodna w użyciu

Dlatego sztuczne sieci neuronowe budujemy tak, żeby ich struktura była wygodna do prześledzenia oraz tania w realizacji. W efekcie są one po pierwsze **plaskie** (a nie trójwymiarowe) i mają narzuconą regularną budowę, w której występują warstwy neuronów o dobrze zdefiniowanym przeznaczeniu, łączone według prostej, chociaż rozrzutnej zasady połączeń typu „każdy z każdym” (porównaj rysunek 2.3).

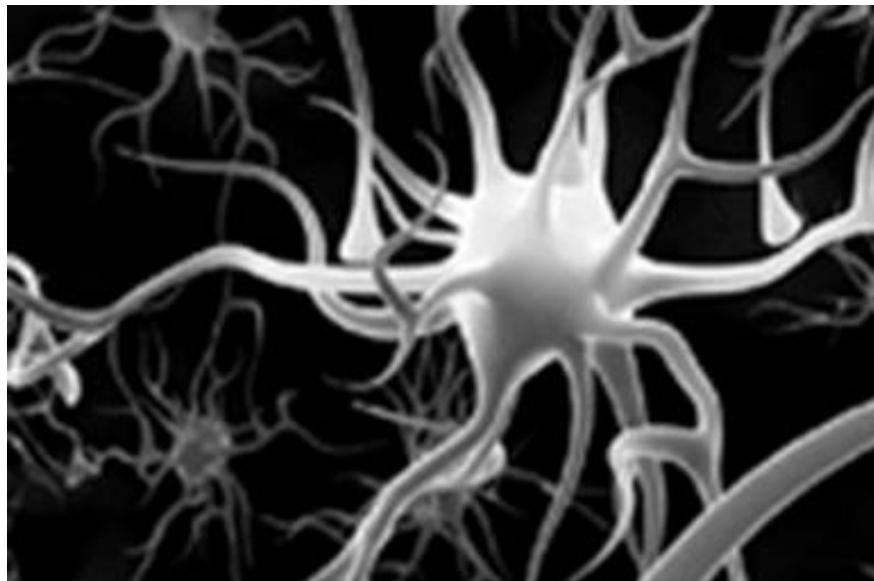
O właściwościach i możliwościach sieci neuronowych decydują trzy czynniki: a) to, z jakich elementów sieć jest zbudowana (czyli jak wyglądają oraz jak działają sztuczne neurony), b) jak te elementy są ze sobą połączone oraz c) w jaki sposób ustala się parametry w sieci za pomocą procesu uczenia. Zajmiemy się teraz tymi czynnikami po kolej.



Rys. 2.3. Schemat praktycznie używanej sieci neuronowej (typu GRNN) pokazuje, że jej budowa jest uproszczona w stosunku do biologicznego oryginału i silnie zracjonalizowana

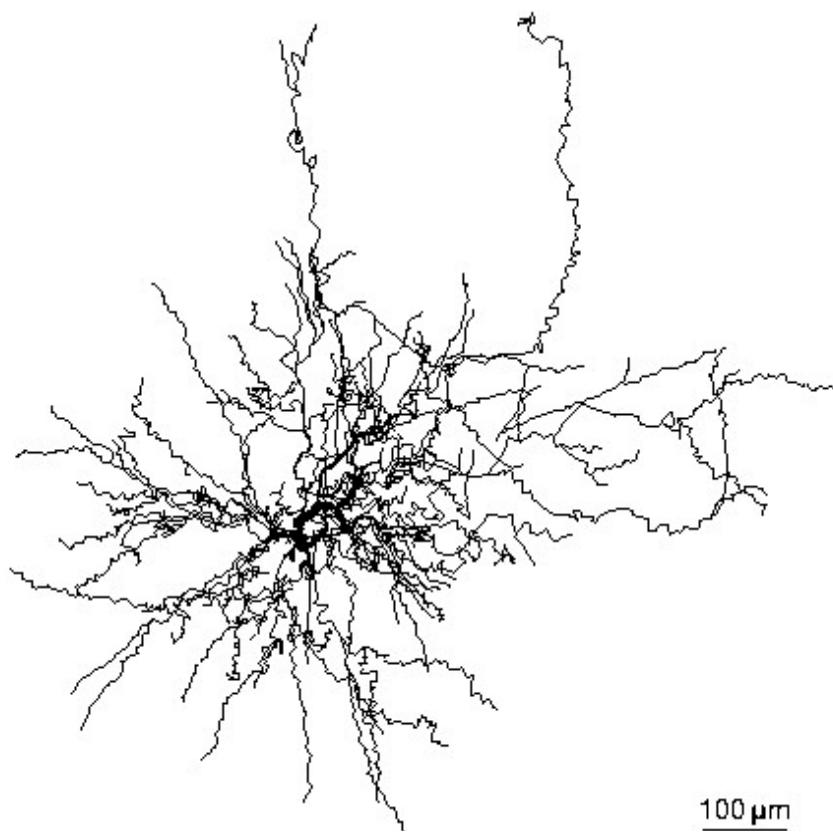
## 2.2. Jak można zrobić sztuczny neuron?

Podstawowym „budulcem”, z którego korzystamy przy tworzeniu sieci neuronowej, są sztuczne neurony. Spróbujmy je teraz poznać nieco dokładniej. W poprzednim rozdziale widziałeś już kilka obrazków ilustrujących kształt typowego biologicznego neuronu, ale dla przypomnienia jeszcze jedna ilustracja nie zaszkodzi, więc zobacz na rysunku 2.4, jak wygląda (w uproszczeniu) przykładowy neuron.



Rys. 2.4. Orientacyjna budowa biologicznej komórki nerwowej (neuronu)

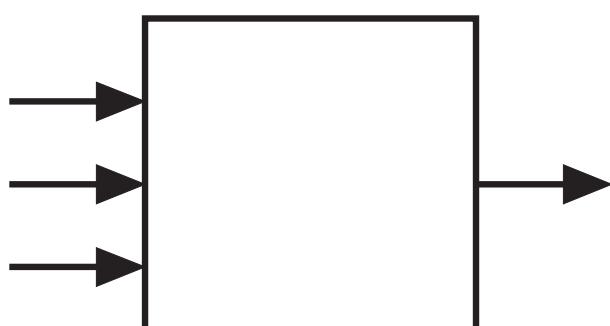
Żebyś nie sądził, że wszystkie prawdziwe neurony wyłącznie tak właśnie wyglądają, na rysunku 2.5 pokazuję Ci jeszcze jeden rysunek prawdziwego biologicznego neuronu, wypreparowanego z kory mózgowej szczura.



Rys. 2.5. Widok preparatu mikroskopowego rzeczywistego neuronu

Trudno na tym rysunku domyślić się, które z licznych widocznych na nim włókien jest **aksonem**, który jest zawsze pojedynczy i jako jedyny wyprowadza sygnały **od** danego neuronu do wszystkich innych, a które pełnią rolę **dendrytów**. Jednak jest to także prawdziwy biologiczny neuron, a więc także i taką komórkę musi dobrze odwzorowywać nasz sztuczny neuron, którym się teraz dokładniej zajmiemy.

Sztuczne neurony budujące wykorzystywane w technice sieci są oczywiście bardzo uproszczonymi modelami komórek nerwowych występujących w przyrodzie. Budowę sztucznego neuronu najlepiej ilustruje schemat przedstawiony na rysunku 2.6. Porównując ten rysunek z rysunkami 2.4 czy 2.5, uświadomisz sobie poglądowo, jak bardzo twórcy sieci neuronowych upraszczają biologiczną rzeczywistość.

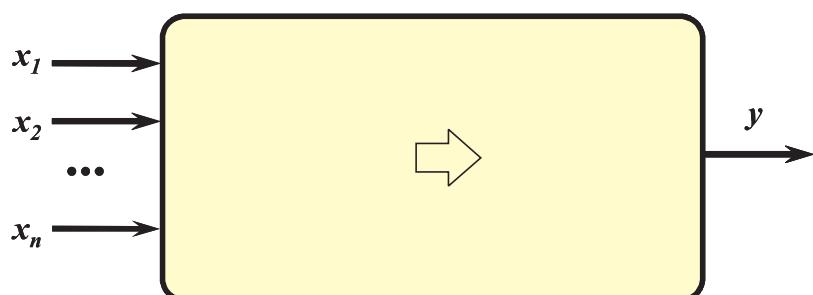


Rys. 2.6. Ogólny schemat sztucznego neuronu pokazuje stopień jego uproszczenia

Jednak mimo tych uproszczeń sztuczne neurony zachowują wszystkie te cechy, które są ważne z punktu widzenia zadań, jakie chcemy im powierzyć w budowanych sieciach, będących narzędziami informatyki, a nie modelami biologii:

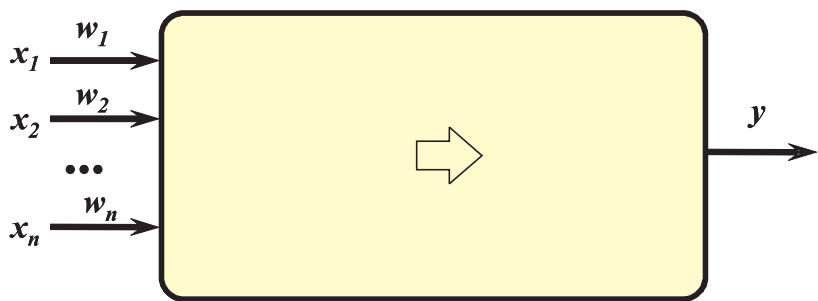
- Po pierwsze – **charakteryzują się one występowaniem wielu wejść i jednego wyjścia.** Sygnały wejściowe  $x_i$  ( $i = 1, 2, \dots, n$ ) oraz sygnał wyjściowy  $y$  mogą przyjmować wyłącznie wartości liczbowe, najczęściej z zakresu od 0 do 1 (czasem także od -1 do +1), natomiast fakt, że w rozwiązywanych przez sieć zadaniach odpowiadają one pewnym informacjom (na przykład na wyjściu decyzji, jaką osobę rozpoznała sieć neuronowa analizująca czyjaś fotografię), jest wynikiem specjalnej **umowy**. Najczęściej konkretne znaczenia przypisuje się do sygnałów wejściowych i wyjściowych sieci w taki sposób, że najistotniejsze jest to, na którym wejściu lub wyjściu określony sygnał się pojawił (każde wejście i każde wyjście związane jest z określonym **znaczeniem** sygnału), zaś dodatkowo stosuje się **skalowanie** sygnałów, tak dobrane, żeby wartości sygnałów, jakie będą w sieci cyrkulowały, nie wykraczały poza ustalony zakres – na przykład od 0 do 1.

- Po drugie – sztuczne neurony wykonują określone czynności na sygnałach, które otrzymują na wejściach, w wyniku czego produkują sygnały (tylko jeden dla każdego pojedynczego neuronu), które są obecne na ich wyjściach i są przesyłane dalej (do innych neuronów albo na wyjście sieci, jako rozwiązywanie postawionego problemu). Zadanie sieci, sprowadzone do funkcjonowania jej podstawowego elementu, jakim jest neuron, polega więc na tym, że przetwarza on informacje wejściowe  $x_i$  na wynik  $y$  stosując reguły wynikające z tego, jak jest zbudowany, oraz z tego, czego został nauczony. Omówione do tej chwili właściwości neuronu przedstawia rysunek 2.7.



Rys. 2.7. Podstawowe sygnały występujące w neuronie

- Po trzecie – neurony mogą się uczyć. Służą do tego współczynniki  $w_i$  nazywane **wagami synaptycznymi**. Jak zapewne pamiętasz z poprzedniego rozdziału – są one odzwierciedleniem dosyć złożonych procesów biochemicznych i bioelektrycznych zachodzących w rzeczywistych synapsach neuronów biologicznych. Z punktu widzenia dalszych rozważań najważniejsze jest to, że wagi synaptyczne mogą podlegać modyfikacjom (czyli mieć zmieniane

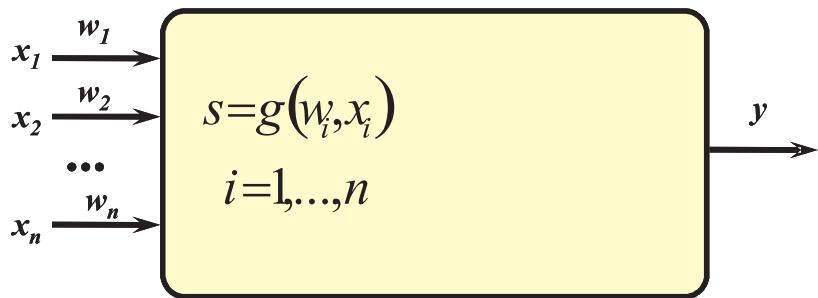


Rys. 2.8. Dodanie do struktury neuronu nastawialnych współczynników wag czyni z niego jednostkę uczącą się

wartości), co stanowi podstawę uczenia sieci. Schemat neuronu zdolnego do uczenia przedstawiono na rysunku 2.8.

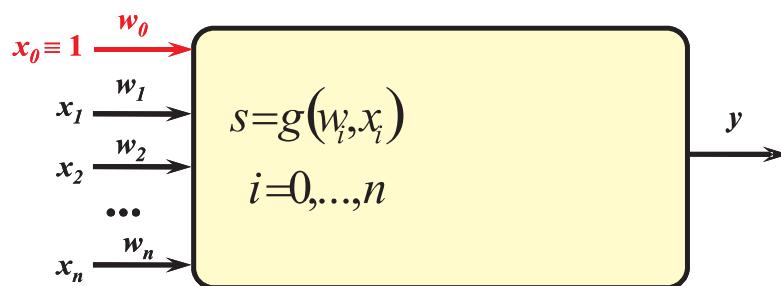
Podsumowując te rozważania można stwierdzić, że sztuczne neurony traktować można jako elementarne **procesory** o następujących właściwościach:

- każdy neuron otrzymuje **wiele sygnałów wejściowych**  $x_i$  i wyznacza na ich podstawie swoją „odpowiedź”  $y$ , to znaczy produkuje **jeden sygnał wyjściowy**;
- z każdym oddzielnym wejściem neuronu związany jest parametr nazywany **wagą** (w literaturze angielskiej *weight*)  $w_i$ . Nazwa ta oznacza, że wyraża on stopień ważności informacji docierających do neuronu tym właśnie wejściem;
- **sygnał** wchodzący określonym wejściem jest najpierw **modyfikowany** z wykorzystaniem wagi danego wejścia. Najczęściej modyfikacja polega na tym, że sygnał jest po prostu **przemnażany** przez wagę danego wejścia, w związku z czym w dalszych obliczeniach uczestniczy już w formie zmodyfikowanej: **wzmocnionej** (gdy waga jest większa od 1) lub **stłumionej** (gdy waga ma wartość mniejszą od 1). Sygnał z danego wejścia może wystąpić nawet w formie **przeciwstawnej** w stosunku do sygnałów z innych wejść, gdy jego waga ma wartość ujemną. Wejścia z ujemnymi wagami są wśród użytkowników sieci neuronowych określane jako tzw. **wejścia hamujące**, natomiast te z wagami dodatnimi są w związku z tym nazywane **wejściami pobudzającymi**;
- sygnały wejściowe (zmodyfikowane przez odpowiednie wagi) są w neuronie **agregowane** (patrz rys. 2.9). Znowu rozważając sieci w ogólny sposób można by było podawać tu różne sposoby agregacji sygnałów wejściowych, jednak najczęściej polega ona na tym, że odpowiednie sygnały są po prostu **sumowane**, dając w wyniku pewien pomocniczy sygnał wewnętrzny, nazywany łącznym pobudzeniem neuronu albo pobudzeniem postsynaptycznym. W literaturze angielskiej dla określenia tego sygnału bywa używany termin *net value*;
- do tak utworzonej sumy sygnałów neuron dodaje niekiedy (nie we wszystkich typach sieci, ale generalnie często) pewien dodatkowy składnik



Rys. 2.9. Agregacja danych wejściowych jako pierwsza z wewnętrznych funkcji neuronu

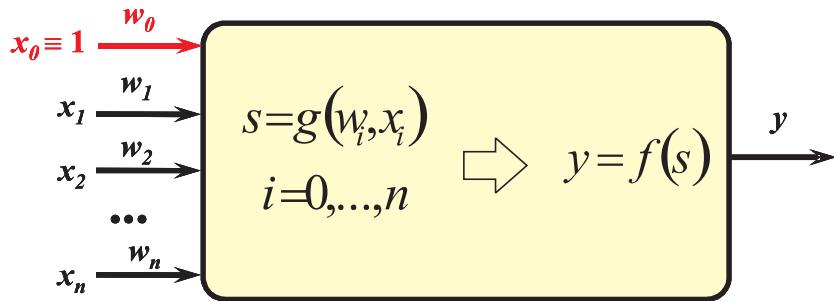
niezależny od sygnałów wejściowych, nazywany **progiem** (w literaturze angielskiej nazywany *bias*). Próg, jeśli jest uwzględniany, także podlega procesowi uczenia, dlatego czasem można sobie wyobrazić, że próg jest dodatkową wagą synaptyczną związaną z wejściem, na które podłączony jest wewnętrzny sygnał o wartości stale wynoszącej 1. Rola progu polega na tym, że dzięki jego obecności właściwości neuronu mogą być kształtowane w trakcie procesu uczenia w sposób znacznie bardziej swobodny (bez jego uwzględnienia charakterystyka funkcji agregacji zawsze musi przechodzić przez początek układu współrzędnych, co stanowi niekiedy uciążliwą „kotwicę”). Schemat neuronu, w którym uwzględniono próg, pokazano na rysunku 2.10;



Rys. 2.10. Zastosowanie dodatkowego parametru, jakim jest próg (*bias*)

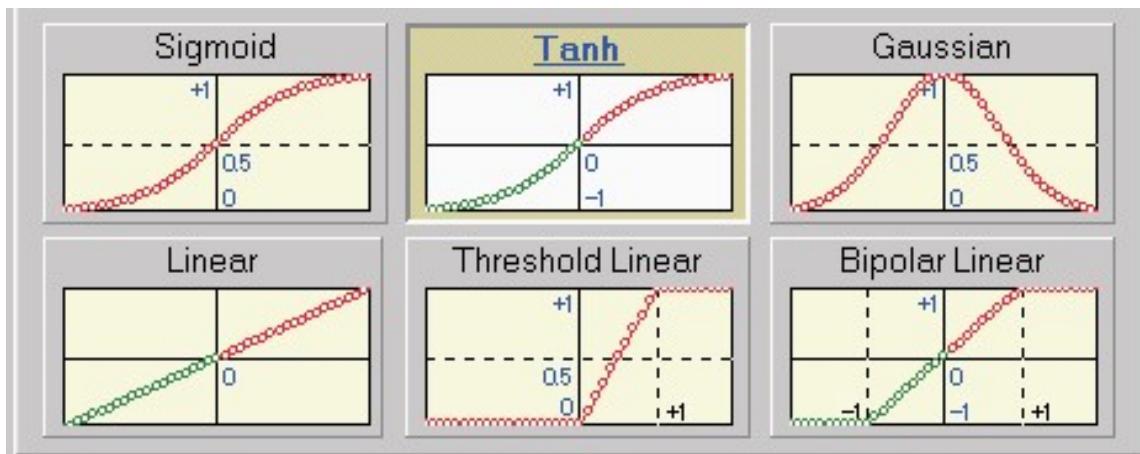
- suma przemnożonych przez wagi sygnałów wewnętrznych z dodanym (ewentualnie) progiem może być niekiedy bezpośrednio przesyłana do jego aksonu i traktowana jako **sygnał wyjściowy** neuronu. W wielu typach sieci to wystarczy. W taki sposób działają tak zwane sieci liniowe (na przykład sieci nazywane **ADALINE** = *ADaptive LINEar*). Natomiast w sieciach o bogatszych możliwościach (na przykład w bardzo popularnych sieciach nazywanych **MLP** od słów *Multi-Layer Perceptron*) sygnał wyjściowy neuronu obliczany jest za pomocą pewnej nieliniowej funkcji. Funkcję tę w całej książce oznaczać będziemy symbolem  $f()$  albo  $\phi()$ . Schemat neuronu zawierającego zarówno agregację sygnałów wejściowych, jak i generację sygnału wyjściowego przedstawia rysunek 2.11;

- funkcja  $\phi()$  zwana jest **charakterystyką neuronu** (w angielskiej literaturze funkcjonuje termin *transfer function*). Znane są różne charakterystyki neuronu, co ilustruje rysunek 2.12. Niektóre z nich są tak wybrane, żeby



Rys. 2.11. Komplet wewnętrznych funkcji neuronu

zachowanie neuronu sztucznego maksymalnie przypominało zachowanie rzeczywistego neuronu biologicznego (funkcja sigmoidalna), ale mogą być one dobrane w taki sposób, żeby zapewniać maksymalną sprawność obliczeń przeprowadzanych przez sieć neuronową (funkcja Gaussa). We wszystkich przypadkach funkcja  $\phi()$  stanowi ważny element pośredniczący między łącznym pobudzeniem neuronu a jego sygnałem wyjściowym;

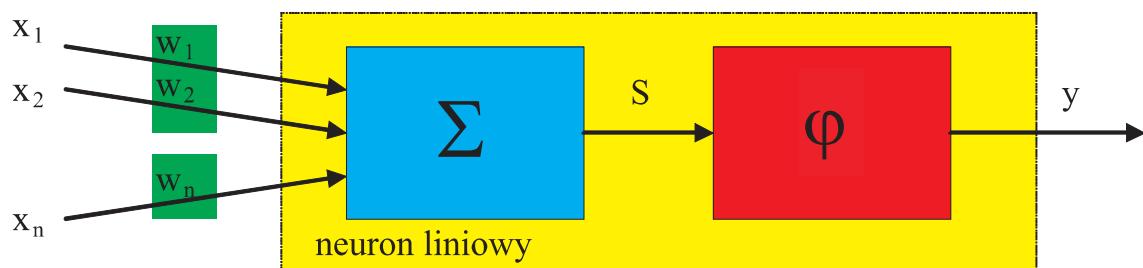


Rys. 2.12. Niektóre częściej używane charakterystyki neuronu

- znajomość sygnałów wejściowych, współczynników wag, sposobu agregacji wejść oraz charakterystyki neuronu, pozwala w każdej chwili jednoznacznie określić jego sygnał wyjściowy, przy czym zwykle zakłada się, że (w odróżnieniu od tego, co dzieje się w rzeczywistych neuronach) proces ten zachodzi **bezzwłocznie**. Dzięki temu w sztucznych sieciach neuronowych zmiany sygnałów wejściowych praktycznie natychmiast uwidaczniane są na wyjściu. Oczywiście, jest to założenie czysto teoretyczne, gdyż nawet przy realizacji elektronicznej konieczny jest pewien czas do tego, żeby po zmianie sygnałów wejściowych odpowiedni układ scalony ustalił właściwą wartość sygnału wyjściowego. Znacznie więcej czasu potrzeba na to, żeby ten sam efekt zrealizować w sieci działającej jako model symulacyjny, bo komputer naśladowujący działanie sieci musi wtedy przeliczyć wszystkie wartości wszyst-

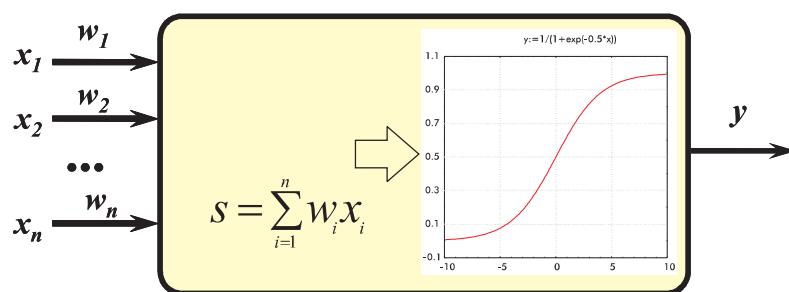
kich sygnałów na wyjściach wszystkich neuronów sieci, co nawet przy bardzo szybkich komputerach może trwać dosyć długo. Mówiąc o bezzwłocznym działaniu neuronów mam na myśli to, że rozważając działanie sieci nie będziemy zwracać uwagi na czynnik, jakim jest czas reakcji neuronu, bo będzie on dla nas nieistotny.

Kompletna opisana struktura pojedynczego neuronu przedstawiona jest na rysunku 2.13.



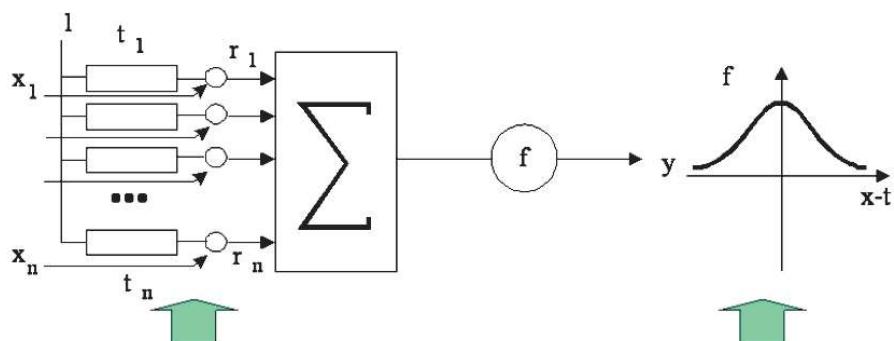
Rys. 2.13. Struktura neuronu jako procesora będącego podstawą budowy sieci neuronowych

Neuron przedstawiony na tym rysunku jest najbardziej typowym „budulcem”, jakiego używa się przy tworzeniu sieci neuronowych. Dokładniej – takim typowym „budulcem” jest neuron sieci określonej jako MLP (*Multi-Layer Perceptron*), którego najważniejsze elementy zebrałem razem i pokazałem na rysunku 2.14. Widać na tym rysunku, że neuron MLP cechuje się funkcją agregacji polegającą na prostym sumowaniu przemnożonych przez „wagi” sygnałów wejściowych oraz korzysta z nieliniowej funkcji przejścia o charakterystycznym kształcie sigmoidy.



Rys. 2.14. Najczęściej spotykany składnik sieci neuronowych – neuron typu MLP

Jednak czasami, w specjalnych celach używane są w sieciach neuronowych tak zwane **neurony radialne**. Mają one nietypową metodę agregacji danych wejściowych, używają też nietypowej charakterystyki (Gaussowskiej) oraz są w nietypowy sposób uczone. Nie chcę w tym momencie rozwijać tematu tych specyficznych neuronów, wykorzystywanych głównie do tworzenia specjalnych sieci określanych jako RBF (od angielskiego *Radial Basis Functions*), ale na rysunku 2.15 przedstawię schemat takiego neuronu radialnego,



Agregacja sygnałów wejściowych w tym typie neuronu polega na obliczaniu **odległości** pomiędzy obecnym wektorem wejściowym  $\mathbf{X}$  a ustalonym podczas uczenia **centroidem** pewnego podzbioru  $T$

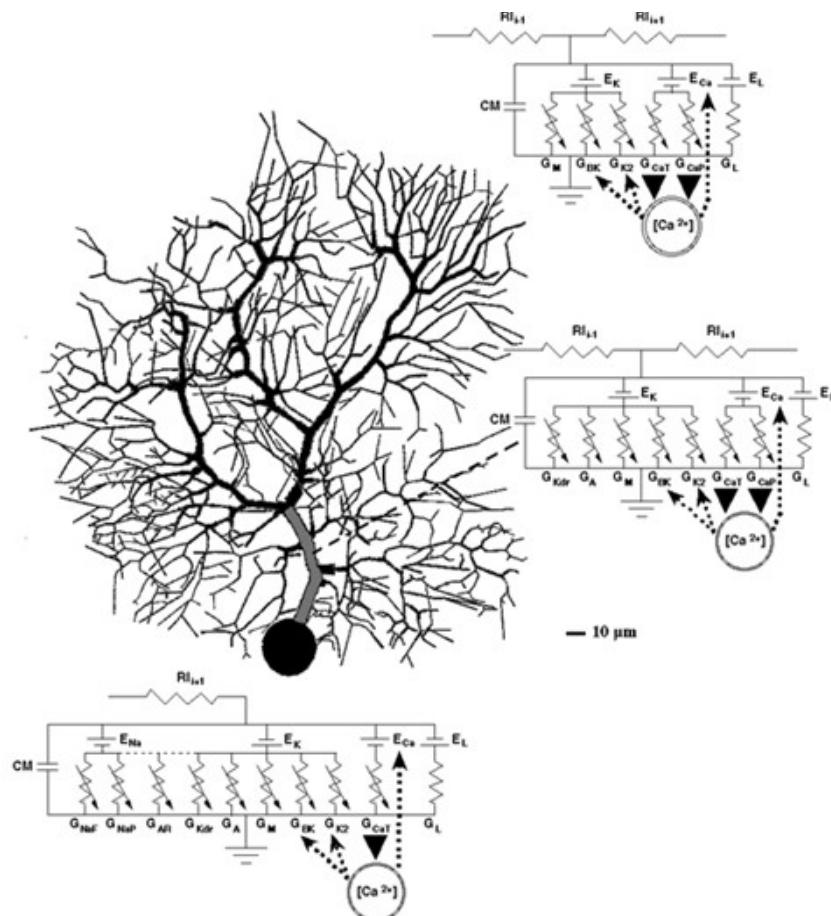
Również nieliniowa funkcja przejścia w tych neuronach ma odmienną formę - „dzwonu” gaussoidy - czyli jest funkcją **niemonotoniczną**

Rys. 2.15. Struktura i osobliwe właściwości neuronu radialnego, oznaczanego też RBF

żeby umożliwić Ci porównanie z wcześniej omówionym typowym neuronem przedstawionym na rysunku 2.14.

### 2.3. Dlaczego nie używamy dokładnego modelu biologicznego neuronu?

Wszystkie sztuczne neurony, sigmoidalne i radialne, opisane w tym rozdziale, a także używane w dalszych częściach tej książki, są **uproszczonymi** modelami rzeczywistych biologicznych neuronów. Stwierdzenie to pojawiało się już kilkakrotnie, teraz jednak chcę Ci pokazać, **jak bardzo uproszczone są sztuczne neurony**. Posłużę się w tym celu przykładem badań prowadzonych przez de Schuttera. Otóż badacz ten przez wiele lat zajmował się tym, żeby maksymalnie wiernie i maksymalnie dokładnie odtworzyć w modelu komputerowym wszystko to, co wiemy na temat struktury i działania (w najdrobniejszych szczegółach!) **jednego** tylko neuronu – konkretnie tak zwanej komórki Purkiniego. Jego model odwoływał się do układów elektrycznych, które zgodnie z badaniami Hodgkina i Huxleya (nagroda Nobla z 1963 roku) modelują bioelektryczną aktywność poszczególnych włókien (dendrytów i aksonu) oraz błony komórkowej ciała neuronu. W badaniach odtworzono z niezwykłą dokładnością kształt rzeczywistej komórki Purkiniego oraz uwzględniono wyniki badań Nehera i Sakmanna (nagroda Nobla z 1991 roku) na temat funkcjonowania tak zwanych kanałów jonowych. Struktura modelowanej komórki oraz schematy elektrycznych układów zastępczych, wykorzystywane w modelu de Schuttera pokazane są na rysunku 2.16.

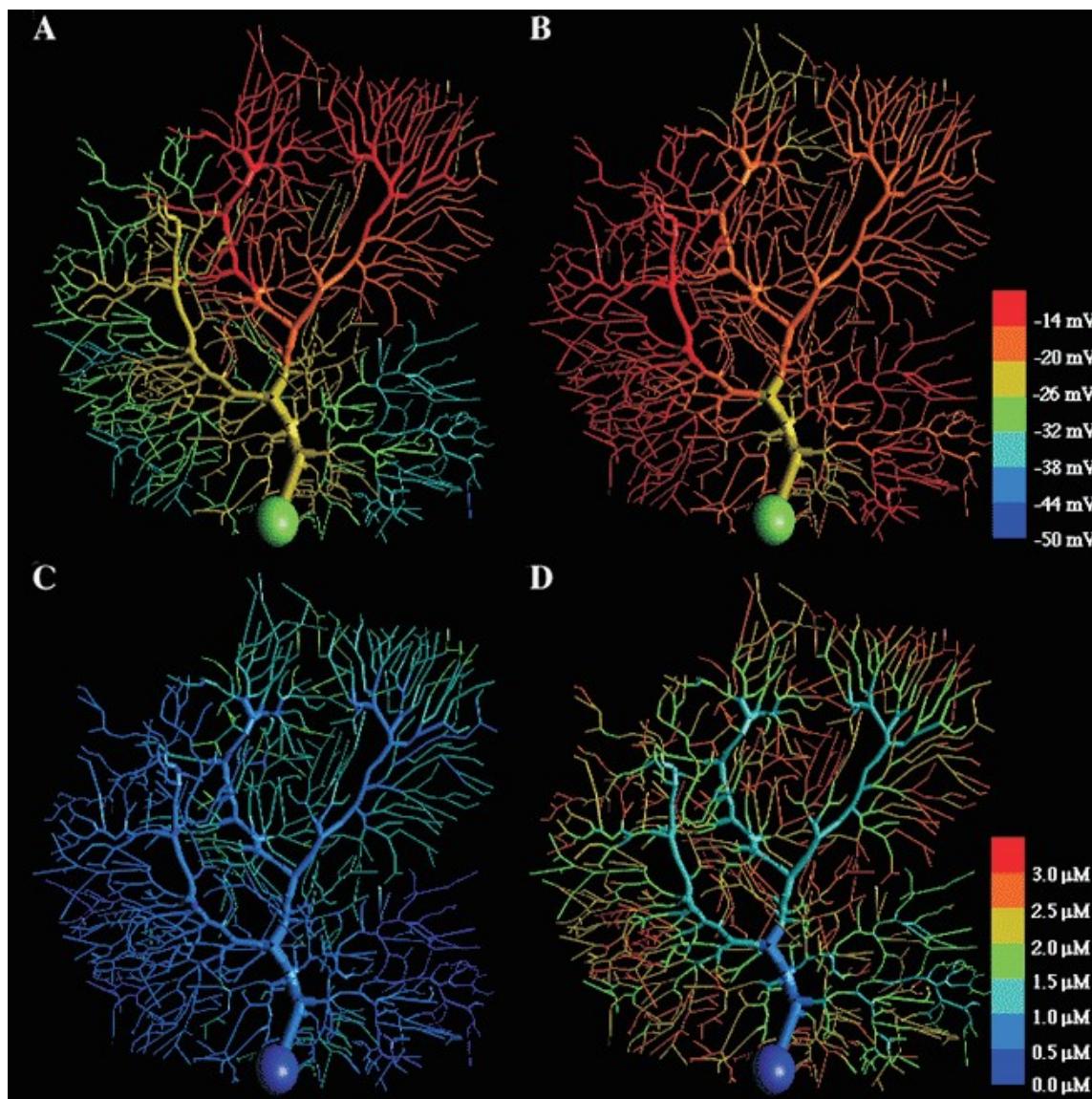


Rys. 2.16. Model neuronu maksymalnie wierny biologicznemu oryginałowi, używany w badaniach de Schuttera

Model, który zbudował de Schutter, okazał się niesłychanie skomplikowany i kosztowny obliczeniowo. Wystarczy powiedzieć, że do zbudowania modelu użycie:

- ⇒ 1600 tzw. ***kompartamentów*** (fragmentów komórki traktowanych jako jednorodne struktury zawierające określone związki chemiczne w określonych stężeniach),
- ⇒ 8021 modeli kanałów jonowych,
- ⇒ 10 typów różnych złożonych opisów matematycznych kanałów jonowych zależnych od napięcia,
- ⇒ **32000 równań różniczkowych (!),**
- ⇒ **19200 parametrów** koniecznych do oszacowania przy dostrajaniu modelu,
- ⇒ dokładnego opisu morfologii komórki zrekonstruowanej za pomocą mikroskopu.

Nic dziwnego, że do zasymulowania kilkunastu sekund „życia” takiej komórki nerwowej konieczne było użycie wielkiego superkomputera, a i tak wy-



Rys. 2.17. Wybrane wyniki uzyskane w badaniach de Schuttera. U góry aktywność elektryczna symulowanej komórki, u dołu zjawiska biochemicalne (przepływ jonów wapnia)

magało to wielu godzin jego nieprzerwanej pracy. Trzeba przyznać, że wyniki tego modelowania są bardzo efektowne. Niektóre z nich przedstawione są na rysunku 2.17.

Jednak wnioski z tych doświadczeń są jednoznaczne: Próba wiernego modelowania struktury i działania rzeczywistego biologicznego neuronu okazała się udana, ale jest to droga zbyt kosztowna, żeby w ten sposób próbować tworzyć **użyteczne praktyczne** sieci neuronowe. Dlatego będziemy od tej chwili już stale używali wyłącznie modeli uproszczonych, oczekując, że mimo tych zastosowanych uproszczeń sieć neuronowa będzie mogła nie tylko skutecznie rozwiązywać różne stawiane jej zadania, ale dodatkowo będzie także zdolna

do tego, że jej zachowania mogą nam nasuwać ciekawe wnioski na temat zachowania ludzkiego (na przykład Twojego!) mózgu. Wkrótce się o tym sam przekonasz!

## 2.4. Jak działa sieć zbudowana ze sztucznych neuronów?

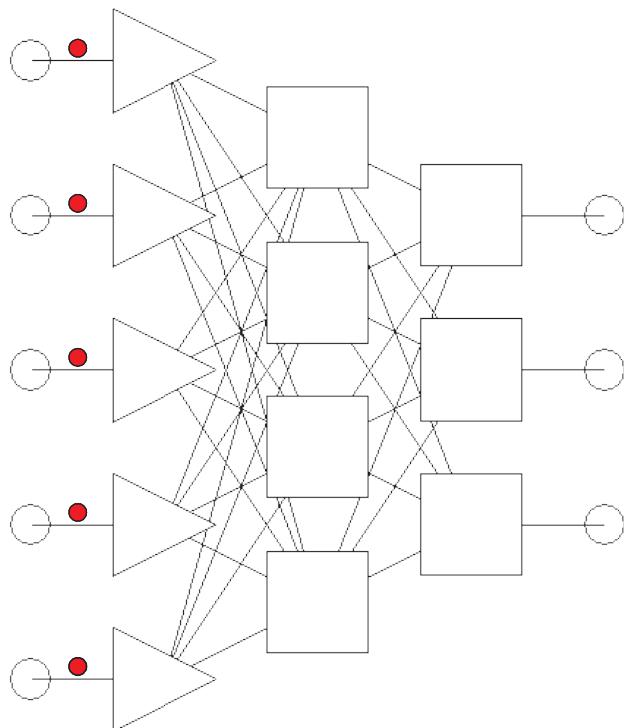
Z przytoczonego wyżej opisu wynika, że każdy neuron dysponuje pewną **wewnętrzną pamięcią** (reprezentowaną przez aktualne wartości wag i progu) oraz pewnymi możliwościami **przetwarzania** wejściowych sygnałów w sygnał wyjściowy. Chociaż możliwości te są dość ograniczone (dzięki temu właśnie neuron jest stosunkowo mało kosztownym procesorem i można zbudować system zawierający setki czy tysiące takich elementów), to jednak okazują się one wystarczające do zbudowania systemów realizujących bardzo złożone zadania przetwarzania danych.

Na skutek tego, że zarówno zasób informacji gromadzonych przez każdy pojedynczy neuron sieci jest ograniczony (no bo jeden neuron ma na ogół niewiele nastawialnych wag), jak i z powodu bardzo ubogich możliwości obliczeniowych takiego pojedynczego neuronu (tylko agregacja sygnałów w wyliczenie sygnału wyjściowego) – **sieć neuronowa z reguły musi zawierać wiele neuronów i może działać wyłącznie jako całość**. Jak z tego wynika, wszystkie omówione w poprzednim rozdziale możliwości i właściwości sieci neuronowych są wynikiem **kolektywnego działania** bardzo wielu połączonych ze sobą elementów (całej sieci, a nie pojedynczych neuronów), w związku z czym uzasadniona jest spotykana niekiedy nazwa **MPP (Massive Parallel Processing)**, jaką wiąże się z całą tą dziedziną informatyki.

Zainteresujmy się na chwilę funkcjonowaniem całej sieci neuronowej. Jak wynika z przytoczonych wyżej uwag, zarówno **program** działania oraz informacje stanowiące **bazę wiedzy**, a także **dane**, na których wykonuje się obliczenia, jak i sam **proces obliczania** – są w sieci całkowicie **rozproszone**. Nie da się wskazać miejsca, w którym zgromadzona jest w niej taka lub inna konkretna informacja, chociaż sieci bywają wykorzystywane jako pamięci, zwłaszcza tzw. pamięci skojarzeniowe i bardzo dobrze spełniają swoje zadania. Podobnie niemożliwe jest zlokalizowanie w określonym miejscu sieci jakiegoś wydzielonego fragmentu wykonywanego przez nią algorytmu, na przykład wskazanie, które elementy sieci odpowiedzialne są za wstępne przetwarzanie i analizę wprowadzanych danych, a które dostarczają końcowego rozwiązania podawanego przez sieć.

Popatrzmy, jak to działa i jaka jest rola poszczególnych składników w funkcjonowaniu całej sieci. W rozważaniach tu prowadzonych założymy, że sieć **ma** już ustalone wszystkie współczynniki wagowe, czyli że proces

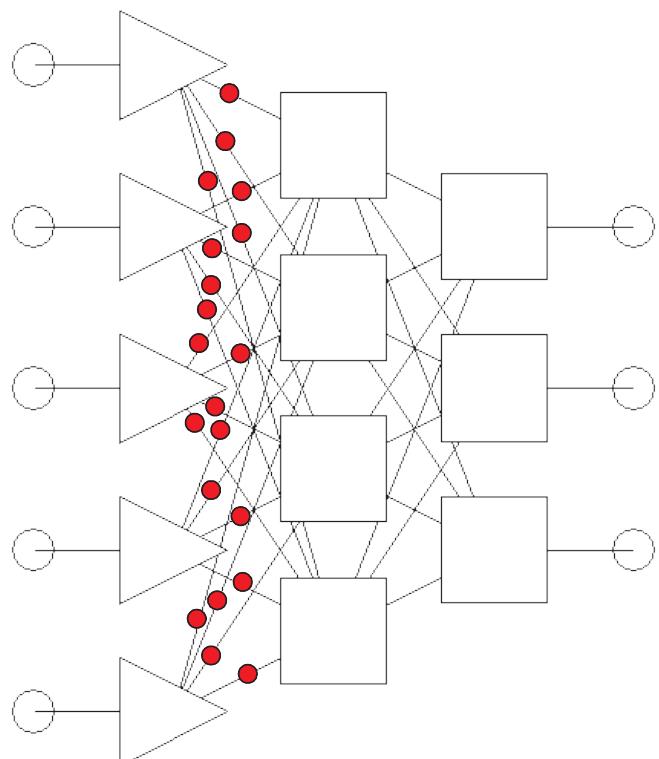
jej uczenia został zakończony. Uczeniem sieci, które jest bardzo ważne, ale i trudniejsze do prześledzenia, zajmiemy się bowiem osobno w następnych rozdziałach. Zaczniemy naszą analizę od momentu, kiedy sieci postawione zostaje nowe zadanie do rozwiązyania. Zadanie to niosą sygnały wejściowe, które pojawiają się na wszystkich wejściach sieci. Na rysunku 2.18 sygnały te symbolizują czerwone punkty ulokowane w odpowiednich miejscach schematu sieci.



Rys. 2.18. Początek działania sieci neuronowej wiąże się z pojawieniem się na jej wejściach sygnałów (czerwone kropki) niosących nowe zadanie do rozwiązania

Sygnały wejściowe trafiają do neuronów warstwy wejściowej, które ich z reguły **nie przetwarzają**, tylko dokonują ich dystrybucji – to znaczy rozmawiają do wszystkich neuronów warstwy ukrytej (rys. 2.19). Na marginesie warto zauważyć, że odmienna rola neuronów warstwy wejściowej, które nie przetwarzają sygnałów, tylko je rozmawiają, akcentowana bywa na rysunkach przedstawiających sieci neuronowe odmiennym kształtem symboli graficznych oznaczających te neurony na schematach (na przykład używany jest trójkąt zamiast kwadratu, jak na przedstawianych tu rysunkach).

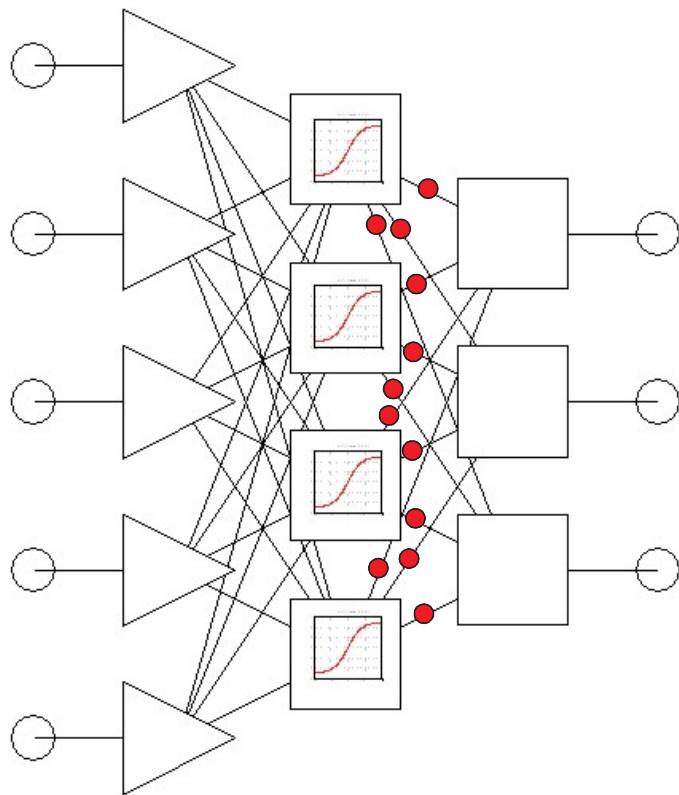
Kolejny etap polega na aktywizacji neuronów warstwy ukrytej. Neurony te wykorzystując swoje wagę (czyli angażując zgromadzoną w nich wiedzę) najpierw modyfikują sygnały wejściowe, następnie agregują je, a potem korzystając ze swoich charakterystyk (na rysunku 2.20 przedstawionych jako funkcje sigmoidalne) wyliczają sygnały wyjściowe, które zostają skierowane do neuronów warstwy wyjściowej. Ten etap przetwarzania informacji w sieci



Rys. 2.19. Sygnały wejściowe (nie przetworzone w żaden sposób w warstwie wejściowej) są rozsyłane do wszystkich neuronów warstwy ukrytej

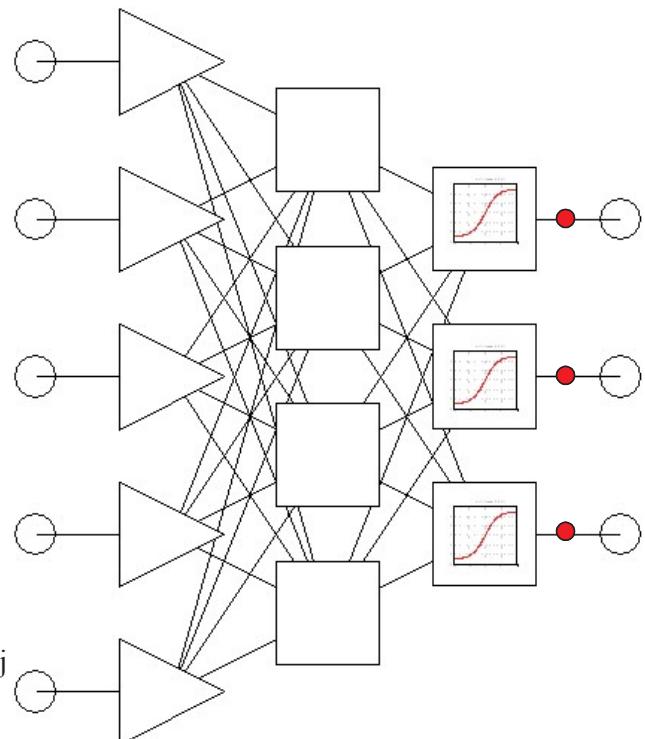
neuronowej ma szczególnie duże znaczenie. Warstwy ukrytej wprawdzie nie widać z zewnątrz sieci (jej sygnałów nie da się zaobserwować ani na wejściu, ani na wyjściu sieci, stąd taka właśnie, a nie inna jej nazwa), ale to tutaj wykonywana jest większa część pracy, związanej z rozwiązywaniem zadania. Najwięcej połączeń i najczęściej związkanych z nimi współczynników wagowych znajduje się zwykle właśnie pomiędzy warstwą wejściową a warstwą ukrytą, stąd można powiedzieć, że większość wiedzy, zgromadzonej w trakcie uczenia, lokuje się właśnie w tej warstwie. Sygnały produkowane przez neurony warstwy ukrytej nie mają żadnej bezpośredniej interpretacji, w odróżnieniu od sygnałów wejściowych i wyjściowych, z których każdy coś konkretnego znaczy w kontekście rozwiązywanego zadania. Jednak posługując się analogią z procesem produkcyjnym można powiedzieć, że neurony warstwy ukrytej wytwarzają „półprodukty”, czyli sygnały w taki sposób charakteryzujące rozwiązywane zadanie, że potem stosunkowo łatwo jest z ich pomocą „zmontować końcowy produkt”, czyli znaleźć ostateczne rozwiązanie w neuronach warstwy wyjściowej (rys. 2.20).

Śledząc dokładniej działanie sieci na tym końcowym etapie rozwiązywania postawionego problemu możemy zauważyć, że neurony warstwy wyjściowej wykorzystują swoje możliwości agregacji sygnałów oraz swoje charakterystyki żeby zbudować końcowe rozwiązanie, podawane na wyjściu sieci (rys. 2.21).



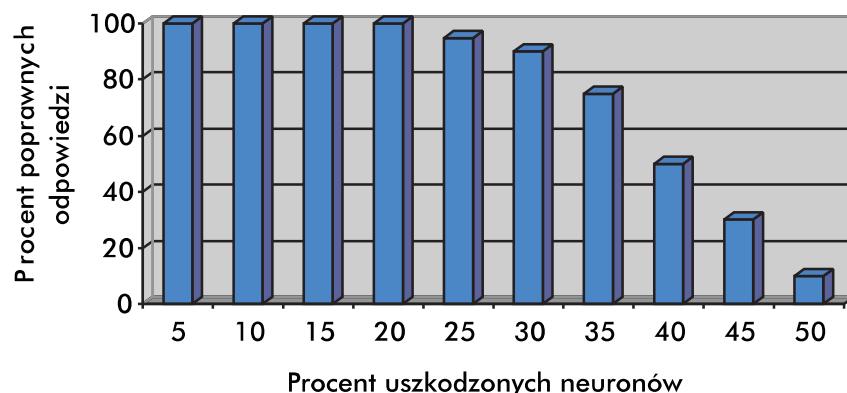
Rys. 2.20. Po przetworzeniu sygnałów przez neurony warstwy ukrytej powstają sygnały pośrednie, kierowane do neuronów warstwy wyjściowej

Powtórzmy jeszcze raz: **sieć działa zawsze jako całość** i wszystkie jej elementy mają swój wkład w realizację wszystkich czynności, które sieć realizuje – podobnie jak się to dzieje przy odtwarzaniu hologramu, gdzie z kaž-



Rys. 2.21. Neurony warstwy wyjściowej korzystają ze wstępnie opracowanej informacji pochodzącej z warstwy ukrytej i obliczają końcowe wyniki, będące rozwiązaniem postawionego zadania

dego kawałka rozbitej płyty fotograficznej można odtworzyć cały obraz sfotografowanego (sholografowanego?) przedmiotu. Jedną z konsekwencji takiego działania sieci jest jej niewiarygodna zdolność do poprawnego działania nawet po uszkodzeniu znacznej części wchodzących w jej skład elementów. Był taki badacz sieci neuronowych (*Frank Rosenblatt*), który najpierw uczył budowane przez siebie sieci neuronowe jakieś umiejętności (na przykład rozpoznawania obrazów liter), a potem egzaminował uszkadzając przy tym celowo coraz większą liczbę ich elementów (sieci były realizowane jako specjalistyczne układy elektroniczne). Okazywało się, że można było uszkodzić sporą część elementów sieci, a ona mimo to prawidłowo wykonywała swoje czynności (rys. 2.22). Uszkodzenie większej liczby neuronów i połączeń powodowało wprawdzie pogorszenie jakości działania sieci, ale polegało ono na tym, że



Rys. 2.22. Sieć neuronowa ma zdumiewającą własność: może działać poprawnie nawet wtedy, gdy uszkodzone zostanie sporo jej elementów składowych!

uszkodzona sieć coraz częściej myliła się przy rozpoznawaniu (wskazywała na przykład D, gdy pokazywano jej literę O) – ale nie odmawiała pracy. Porównaj to zachowanie sieci ze znanim Ci faktem, że w większości innych urządzeń elektronicznych (komputer, telewizor) wystarczy uszkodzenie jednego ważnego elementu, by system przestał działać jako całość, a następnie weź pod uwagę znany fakt, że w mózgu dorosłego człowieka **codziennie** ginie (z różnych względów) kilka tysięcy neuronów – a jednak nasz mózg jako całość działa niezawodnie przez wiele lat.

## 2.5. Jak struktura sieci neuronowej wpływa na to, co sieć może zrobić?

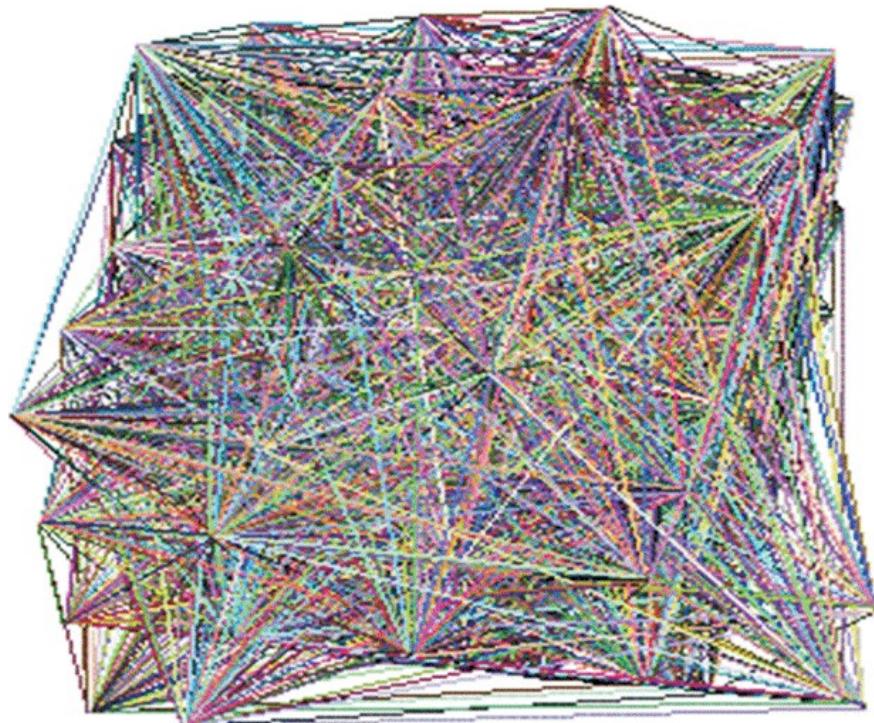
Rozważmy teraz związek, jaki zachodzi pomiędzy strukturą sieci a zadaniami, które może ona wykonywać. Jak już wiesz, z neuronów o właściwościach omówionych w poprzednim podrozdziale tworzona jest sieć. Struktu-

ra tej sieci powstaje w ten sposób, że wyjścia jednych neuronów łączy się (według wybranego schematu) z wejściami innych, tworząc łącznie system zdolny do równoległego, w pełni współbieżnego przetwarzania różnych informacji. Z powodów, o których także była już mowa, wybieramy zwykle sieć o strukturze warstwowej, a połączenia pomiędzy neuronami poszczególnych warstw kształtuje przyjmując zasadę połączeń typu „każdy z każdym”. Oczywiście, konkretna topologia sieci, to znaczy głównie liczba wybranych neuronów w poszczególnych warstwach, powinna wynikać z rodzaju zadania, jakie zamierzamy sieci postawić. W teorii reguła jest dosyć prosta: Im bardziej skomplikowane zadanie, tym więcej neuronów musi mieć sieć, żeby go rozwiązać, bo sieć mająca większą liczbę neuronów jest po prostu intelligentniejsza. W praktyce jednak nie jest to wcale tak jednoznaczne, jak by się mogło wydawać.

W bogatej literaturze dotyczącej sieci znaleźć bowiem można liczne prace, w których wykazano, że w istocie **decyzje dotyczące struktury sieci wpływają na jej zachowanie znacznie słabiej, niż by można było oczekiwąć**. To paradoksalne stwierdzenie wynika z faktu, że zachowanie sieci w zasadniczy sposób determinowane jest przez proces jej **uczenia**, a nie przez **strukturę** czy liczbę użytych do jej budowy elementów. Oznacza to, że sieć mająca zdecydowanie gorszą strukturę może znacznie skuteczniej rozwiązywać postawione zadania (gdy zostanie dobrze nauczona) niż sieć o optymalnie dobranej strukturze, ale źle trenowana. Znane są doświadczenia, w których strukturę sieci wybierano całkowicie **przypadkowo** (ustalając na drodze losowania, które elementy należy ze sobą połączyć i w jaki sposób), a sieć mimo to zdolna była do rozwiązywania stawianych jej trudnych zadań!

Przyjrzymy się uważniej konsekwencjom tego ostatniego stwierdzenia, są one bowiem dosyć ważne i ciekawe. Skoro sieć mogła osiągnąć prawidłowe wyniki, chociaż jej strukturę zaprojektowano całkowicie w sposób przypadkowy, to oznacza to, że proces uczenia mógł tak każdorazowo dostosować jej parametry do wymaganych operacji, wynikających z realizacji ustalonego algorytmu, że proces rozwiązywania zadania przebiegał poprawnie mimo całkowicie **losowej** struktury sieci. Te doświadczenia, wykonane po raz pierwszy przez wspomnianego już wyżej *Franka Rosenblatta* na początku lat 70., były bardzo efektowne: badacz rzucał kostką albo wyciągał losy – i w zależności od tego, co mu z tych losowań wychodziło, łączył pewne elementy sieci ze sobą albo nie. Powstająca struktura połączeń była całkowicie **chaotyczna** (patrz rysunek 2.23) – a jednak sieć po procesie uczenia potrafiła bardzo sensownie wykonywać stawiane jej zadań.

Wyniki tych badań, relacjonowane przez Rosenblatta w jego publikacjach, były tak zadziwiające, że początkowo uczeni nie wierzyli, że coś takiego jest



Rys. 2.23. Struktura połączeń sieci, której elementy były ze sobą łączone z wykorzystaniem reguł losowych. Zdziwiające jest to, że taka sieć może (po procesie uczenia) wykazywać skuteczne i celowe działanie

w ogóle możliwe. Jednak potem doświadczenia te były wielokrotnie powtarzane (między innymi w ZSRR akademik Głuszkow specjalnie w tym celu zbudował sieć neuronową, którą nazwał *Alfa*) i dowiodły, że sieć o przypadkowych połączeniach może nauczyć się poprawnego rozwiązywania zadań, chociaż oczywiście proces uczenia takiej sieci jest dłuższy i trudniejszy niż takiej sieci, której struktura sensownie nawiązuje do zadania, które trzeba rozwiązać.

Co ciekawe, wynikami ogłoszonymi przez Rosenblatta zainteresowali się także filozofowie, którzy uznali, że w ten oto sposób została udowodniona pewna teza, pochodząca jeszcze od Arystotelesa i rozwinięta potem przez Lockea. Chodzi o koncepcję znaną w filozofii pod nazwą *tabula rasa* – umysłu rodzącego się jako pusta, nie zapisana karta, zapełniana dopiero w trakcie nauki i gromadzenia doświadczeń. Rosenblatt dowódł, że ta koncepcja jest technicznie możliwa – przynajmniej w formie sieci neuronowej. Odrębnym zagadnieniem jest próba odpowiedzi na pytanie, czy tak właśnie jest z realnym umysłem konkretnego człowieka? Czy istotnie, jak utrzymywał *Locke*, wrodzone uzdolnienia są niczym, a zdobyta w procesie nauki wiedza – wszystkim?

Nie wiemy tego na pewno, ale chyba jednak tak nie jest. Natomiast wieemy z całą pewnością, że sieci neuronowe mogą całą swoją wiedzę zyskiwać

wyłącznie w trakcie nauki i nie muszą mieć z góry zadanej, dopasowanej do stawianych im zadań, jakiekolwiek precyzyjnie określonej struktury. Oczywiście, sieć musi mieć wystarczający stopień złożoności, żeby w jej strukturze można było w toku uczenia „wykryształzować” potrzebne połączenia i struktury. Zbyt mała sieć nie jest w stanie nauczyć się niczego, gdyż jej „potencjał intelektualny” na to nie pozwala – rzecz jednak nie w strukturze, a w liczbie elementów. Szczura nikt nie uczy teorii względności, chociaż można go wytresować w rozpoznawaniu drogi wewnątrz skomplikowanego labiryntu. Podobnie nikt się nie rodzi „zaprogramowany” do tego, by być wyłącznie genialnym chirurgiem lub koniecznie tylko budowniczym mostów (o tym decydują specjalistyczne studia) – chociaż niektórym ludziom wystarcza intelektu zaledwie do tego, by ładować piasek na ciężarówkę, a i to pod nadzorem. Tak już jest i żadne górnolotne frazesy na temat równości nie są tego w stanie zmienić. Jedni mają wystarczające zasoby intelektualne, inni nie – podobnie jak jedni mają smukłą sylwetkę i okazałą muskulaturę, a inni wyglądają... jakby długo siedzieli przed monitorem komputera.

W przypadku sieci sytuacja jest podobna – nie można spowodować, by sieć z góry miała jakieś szczególne uzdolnienia, można jednak łatwo wyprodukować cybernetycznego kretyna, który nigdy się niczego nie nauczy, bo ma za małe możliwości. Struktura sieci może więc być dowolna, byleby była dostatecznie duża. Wkrótce dowiemy się także, że nie powinna być za duża, bo to także szkodzi – ale o tym porozmawiamy dokładniej za chwilę.

## 2.6. Jak mądrze wybierać strukturę sieci?

Niezależnie od przytoczonych uwag, wskazujących na możliwość osiągnięcia sukcesu poprzez nauczenie określonego działania sieci o strukturze niekoniecznie optymalnie dopasowanej do rozwiązywanego zadania – **jakąś** strukturę trzeba sieci nadać. Ponadto łatwo wykazać, że wybranie na początku **rozsądnej** struktury, dobrze dopasowanej do specyfiki rozwiązywanego zadania, może w istotny sposób skracać czas uczenia i polepszać jego końcowe wyniki. Dlatego pewne uwagi na temat wyboru struktury sieci muszę Ci tu przedstawić, chociaż z pewnością nie będą to uniwersalne recepty na wszystkie możliwe bolączki. Czuję się zobowiązany do podania Ci tutaj kilku wskazówek, ponieważ wszyscy wiemy, jak trudne rozterki wiążą się czasem z wyborem **dowolnego**, nie narzuconego z góry rozwiązania. Postawienie konstruktora sieci w sytuacji, kiedy może przyjąć dowolną jej organizację, jest podobne do dylematów początkujących informatyków, z zakłopotaniem wpatrujących się w pojawiający się niekiedy na ekranie komunikat systemu: *Press any key...*

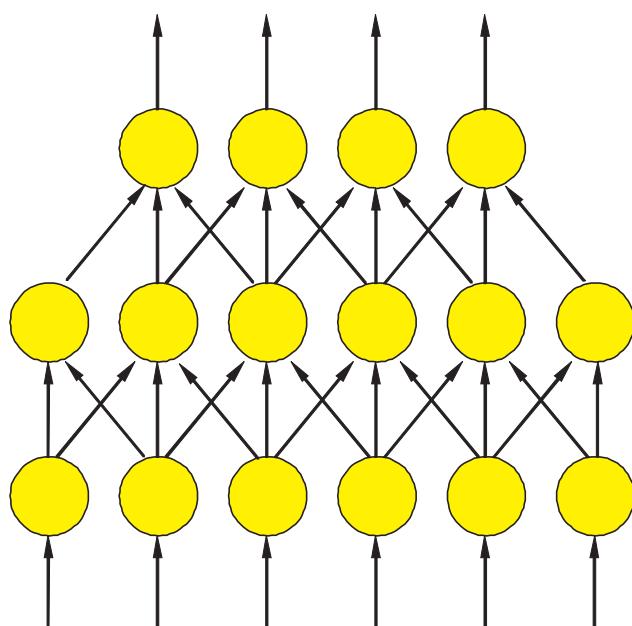
**Który** to jest ten **dowolny** klawisz, który mam nacisnąć?!

Można się z tego śmiać, ale dla mnie podobnie brzmi często spotykane i moich studentów i doktorantów, pełne rozpaczy pytanie: no dobrze, ale **jaka** to jest ta **dowolna** struktura sieci?

Powiem więc teraz kilka słów o możliwych i często spotykanych strukturach sieci, wyraźnie akcentując, że podane niżej informacje i propozycje nie wyczerpują wszystkich możliwości, przeciwnie – każdy badacz może i powinien być tu swoistym Demiurgiem, twórcą i kreatorem nowych bytów, gdyż właściwości sieci o różnych strukturach nie są jeszcze dostatecznie poznane i to jest praca, przy której przyda się każda para... półkul mózgowych.

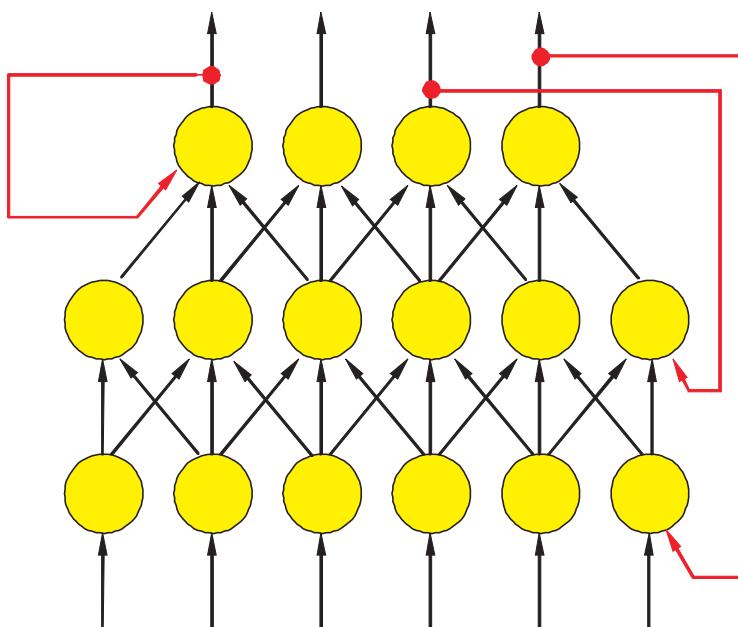
Najpierw dokonam pewnego podziału struktur często stosowanych sieci neuronowych na dwie ważne klasy: z jednej strony rozważać będziemy struktury nie zawierające sprzężeń zwrotnych, a z drugiej struktury, które takie sprzężenia zawierają. Pierwsze z wymienionych sieci określane są często terminem **feedforward**, który jest wprawdzie obco brzmiący i na pozór trochę tajemniczy, ale wygodniejszy od terminu opisowego, jaki by można nadać tym sieciom w naszym języku: „*sieci z jednokierunkowym przepływem sygnału*”. Drugie sieci mogą zawierać sprzężenia zwrotne, w których sygnały mogą krążyć dowolnie długo i dlatego bywają nazywane sieciami **rekurencyjnymi**.

- Sieci *feedforward* to struktury, w których istnieje ściśle określony kierunek przepływu sygnałów – od pewnego ustalonego wejścia, na którym podaje się sieci sygnały będące danymi wejściowymi, precyzującymi zadania, które mają być rozwiązywane – do wyjścia, na którym sieć podaje ustalone rozwiązanie (patrz rysunek 2.24). Takie sieci są najczęściej stosowane i najbardziej użyteczne. Ich obszerniejsze omówienie stanowić będzie treść dalszej części tego rozdziału i kilku następnych.



Rys. 2.24. Przykładowa struktura sieci *feedforward*. Neurony, symbolizowane tu przez żółte kółka, łączone są w taki sposób, że przepływ sygnałów możliwy jest wyłącznie od wejścia do wyjścia

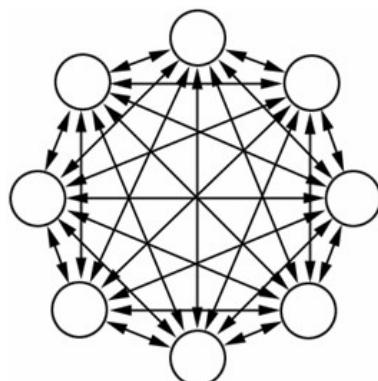
- Sieci *rekurencyjne* cechują się natomiast tym, że neurony tworzą sprzężenia zwrotne, liczne i skomplikowane zamknięte pętle, w których impulsy mogą dłucho krążyć i zmieniać się, zanim sieć osiągnie pewien stan ustalony – o ile go w ogóle osiągnie (rys. 2.25).



Rys. 2.25. Przykładowa struktura sieci rekurencyjnej. Wyróżniono czerwonym kolorem połączenia będące sprzężeniami zwrotnymi, powodujące, że sieć stała się siecią rekurencyjną

Analiza właściwości i możliwości sieci rekurencyjnych jest znacznie bardziej złożona niż w przypadku sieci feedforward, ale też możliwości obliczeniowe tych sieci są fascynująco odmienne od możliwości innych typów sieci – są one na przykład zdolne do znajdowania rozwiązań problemów typu optymalizacyjnego – szukania najlepszych rozwiązań pewnych klas zadań, czego sieci feedforward robić z reguły nie potrafią.

- Wśród sieci rekurencyjnych szczególne miejsce zajmują sieci zвязane z nazwiskiem **Hopfielda**, które mają wyjątkowo dużo sprzężeń zwrotnych. Prawdę mówiąc, w tych sieciach **wszystkie** neurony są sprzężone ze sobą na zasadzie sprzężenia zwrotnego i nie ma w tych sieciach żadnych innych połączeń poza sprzężeniami zwrotnymi (rys. 2.26).



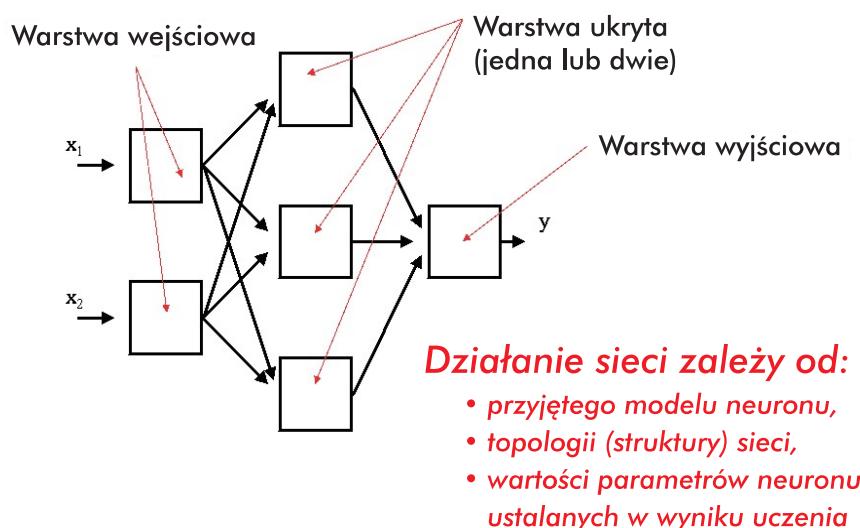
Rys. 2.26. Sieć Hopfielda, w której wszystkie neurony są ze sobą nawzajem powiązane na zasadzie sprzężeń zwrotnych

Swojego czasu prawdziwą sensacją było uzyskanie za pomocą sieci Hopfielda (tej „krańcowo rekurencyjnej”) rozwiązania słynnego problemu komiwojażera, co otworzyło dla tych sieci obszerną i ważną klasę problemów obliczeniowych *NP-zupełnych*, o czym jednak opowiem Ci dokładniej przy innej okazji. Jednak mimo tego sensacyjnego osiągnięcia sieci Hopfielda nie stały się tak popularne, jak inne rodzaje sieci, więc wzmiankujemy o nich dopiero na końcu książki, w rozdziale 11.

Ponieważ budowa sieci ze sprzężeniami zwrotnymi jest bez wątpienia zadaniem trudniejszym i bardziej skomplikowanym niż korzystanie z sieci *feedforward*, a ponadto trudniej jest zapanować nad siecią, w której kłębi się parę tysięcy równoległych, dynamicznych procesów, niż nad siecią, w której sygnały grzecznie i spokojnie przepływają od wejścia na wyjście, dlatego warto **zacząć** znajomość z sieciami neuronowymi właśnie od sieci z jednokierunkowym przepływem sygnałów, przechodząc potem stopniowo i powoli do sieci rekurencyjnych. Jeśli słyszałeś już o najsłynniejszej z nich, sieci Hopfielda i pragniesz z nią zawiążeć znajomość, to albo porzuć czytanie kolejnych rozdziałów i od razu zacznij czytać to, co przygotowałem dla Ciebie w 11. rozdziale, albo (co zdecydowanie polecam) czytaj systematycznie rozdział po rozdziale – ale wtedy musisz się uzbroić w cierpliwość.

Skupiając obecnie uwagę na sieciach feedforward możemy stwierdzić, że do opisania ich struktury stosuje się wygodny i uniwersalny **model warstwowy**. W modelu tym zakłada się, że neurony zgrupowane są w pewne zespoły (warstwy), tak zorganizowane, że główne połączenia i związane z nimi przepływy sygnałów odbywają się pomiędzy elementami sąsiednich warstw. Struktura ta była już omawiana, także i w tym rozdziale, ale nie zaszkodzi przyjrzeć się jej jeszcze raz (rys. 2.27).

Rysunek istotny



Rys. 2.27. Schematyczna budowa najprostszej wielowarstwowej sieci neuronowej

Wspominałem już o tym w poprzednim rozdziale, ale warto może powtórzyć, że połączenia między neuronami sąsiednich warstw mogą być kształtowane na wiele sposobów (wedle uznania twórcy sieci), jednak najczęściej korzysta się ze schematu połączeń typu „każdy z każdym”, licząc na to, że proces uczenia doprowadzi do samorzutnego „wykryształzowania się” potrzebnego zbioru połączeń – po prostu na wejściach, które okażą się zbyteczne z punktu widzenia rozwiązywanego zadania, proces uczenia ustawi współczynniki (wagi) wynoszące zero, co w praktyce przerwie te niepotrzebne połączenia.

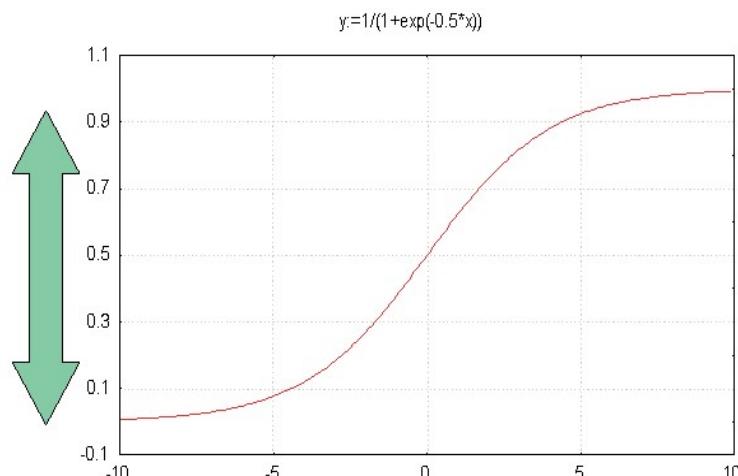
## 2.7. Jakimi informacjami karmić sieci?

Wśród warstw neuronów budujących sieć neuronową pierwsza powinna być omówiona **warstwa wejściowa**. Warstwa ta otrzymuje dane z zewnątrz sieci (tą drogą wprowadzane są zadania podlegające rozwiązywaniu). Przy projektowaniu tej warstwy twórca sieci ma tu ułatwioną decyzję – liczba elementów tej warstwy jest bowiem ściśle zdeterminowana przez liczbę danych wejściowych, które trzeba brać pod uwagę przy rozwiązywaniu określonego zadania. Inna rzecz, że czasem samo tylko zdecydowanie się, ile i jakich danych należy wprowadzać do sieci, by poradziła sobie ona ze stawianym zadaniem, nie jest sprawą łatwą. Na przykład, w zadaniu prognozowania kursów akcji na giełdzie – wiadomo, że niektórzy badacze osiągają tu bardzo zachęcające wyniki, przynoszące bardzo duże korzyści inwestorom, którzy w oparciu o produkowane przez sieć prognozy podejmują decyzje o zakupie lub sprzedaży określonych walorów. Jednak publikacje, jakie się na ten temat pojawiają, są bardzo powściągliwe w ujawnianiu, jakie dane stanowiły punkt wyjścia prowadzonych obliczeń. Owszem, mówi się o tym, że stosowano sieć, że ją uczoно (podaje się nawet algorytm uczenia), podaje się wyniki (jak wiele zyskano dzięki trafnym inwestycjom, jak dokładnie sieć prognozowała zmiany kursów akcji – tu możliwe są bardzo ładne wykresy linii rzeczywistych zmian i linii prognoz), natomiast w odniesieniu do danych wejściowych mówi się tylko, że podawano informacje dotyczące wcześniejszych zmian notowań akcji oraz wyniki analiz finansowych notowanych na giełdzie spółek. Jak te dane przygotowano, które wykorzystano i w jakim stopniu – jakoś autorzy zapominają napisać. Takie roztrzepane gapy!

Z wprowadzeniem danych do sieci neuronowej (a także z uzyskiwaniem od sieci rozwiązań, o czym będzie mowa w kolejnym rozdziale) związana jest jednak pewna subtelność, na którą warto zwrócić uwagę. Otóż neurony dysponują wprawdzie możliwością dostarczania rozwiązania w postaci wartości liczbowej, jednak wartość ta podlega dość istotnym ograniczeniom. Na przykład w większości implementacji sieci sygnały wyjściowe wszystkich

neuronów mogą przyjmować wartości z przedziału od 0 do 1 (lub – co bywa korzystniejsze – od -1 do 1), zatem jeśli potrzebne nam wyniki mają mieć wartości z innego przedziału – konieczne jest pewne **skalowanie** (rys. 2.28).

**cel skalowania:**  
**dopasowanie zakresu wartości zmiennej do charakterystyki nauronu**



Rys. 2.28. Zakres wartości rzeczywistej zmiennej, która ma się pojawić na wejściu lub na wyjściu sieci neuronowej, musi być przeskalowany tak, aby mieścił się w przedziale dozwolonym dla neuronu

Problem skalowania danych wejściowych jest w istocie mniej krytyczny niż w przypadku danych wyjściowych (o czym będzie mowa w następnym podrozdziale), bo do wejściowych neuronów można w zasadzie „wepchnąć” na ich wejściach właściwie dowolny sygnał o dowolnej wartości. Na wyjściu jest inaczej, bo neuron nie potrafi wyprodukować innego sygnału niż taki, jaki pozwala mu wytworzyć jego charakterystyka. Jednak dla zachowania jednorodności interpretacji wszystkich cyrkulujących w sieci sygnałów oraz związanych z nimi wag skalowanie wartości wejściowych wykonywane jest prawie zawsze, co ma też dodatkową zaletę, że przy okazji niejako dokonywana jest tzw. **normalizacja** zmiennych wejściowych.

Wyraz „normalizacja” może brzmieć groźnie i sprawiać wrażenie czegoś bardzo skomplikowanego, ale w istocie chodzi o to, żeby zagwarantować sieci „równouprawnienie” jej wszystkich sygnałów wejściowych. Problem polega na tym, że niektóre zmienne wejściowe, niosące informacje istotne dla rozwiązywanego zadania, mogą przyjmować (z samej swojej natury) niewielkie wartości, podczas gdy inne, wcale nie bardziej ważne, mogą przyjmować wartości bardzo duże. Na przykład, w sieci neuronowej, która powinna pomagać lekarzowi w rozpoznaniu choroby, możemy mieć na jednym wejściu podaną ciepłość ciała pacjenta, a na drugim liczbę erytrocytów (czerwonych krwinek) ustaloną w trakcie analizy jego krwi. Obie te informacje są ważne i o poprawnym rozpoznaniu raz może przesądzić analiza jednej z nich, a innym razem tej

drugiej. Jednak temperatura ciała człowieka wyraża się niewielką liczbą (jak wiesz, u zdrowego człowieka jest to 36,6 stopni Celsjusza) i nawet niewielkie jej zmiany (o kilka stopni w górę lub w dół) mogą znamionować bardzo poważną chorobę. Tymczasem liczba czerwonych krwinek (oznaczana w jednym mililitrze krwi) wynosi około 5 milionów i jej zmiana (nawet o milion) wcale nie jest niczym szczególnie niepokojącym. Gdyby nie przeprowadzono skalowania, to neuron „widzący” na jednym swoim wejściu ogromne wartości liczbowe związane z reprezentacją danych dotyczących analizy krwi – tylko od nich uzależniałby swoje działanie, ignorując całkowicie swoje drugie wejście (to dotyczące temperatury). Dzięki normalizacji obie zmienne stają się w pełni „równouprawnione”, a działanie sieci staje się skuteczniejsze.

## 2.8. Jak wyjaśnić sieci, skąd pochodzi krowa?

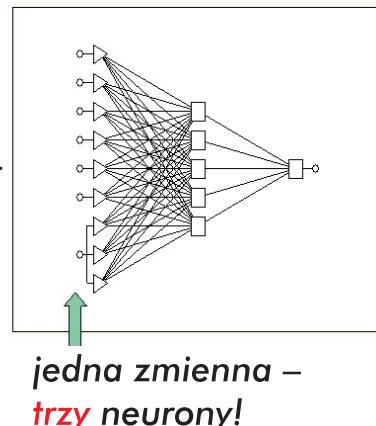
Kolejny problem jest trudniejszy. Dane dostarczane do sieci neuronowej (lub pozyskiwane z niej na zasadzie odczytania potrzebnego rozwiązania) nie zawsze mają wyłącznie charakter liczbowy – i to stanowi źródło poważnych komplikacji. Niestety, świat jest tak urządżony, że nie wszystko da się zmierzyć i wyrazić ilościowo. Wiele danych, którymi chcemy się posłużyć na wejściu lub na wyjściu sieci neuronowej, ma charakter jakościowy, opisowy, albo – jak to się najczęściej nazywa – *nominalny*. To znaczy ich **wartościami** są pewne **nazwy**, a nie **liczby**. Żeby to wyjaśnić, wygodnie będzie odwołać się do przykładu. Wyobraź sobie, że masz do rozwiązania zadanie, w którym sieć neuronowa rozpoznaje, czy pewne zwierzęta mogą być niebezpieczne dla człowieka, czy nie (niebawem będziesz podobne zadania sam rozwiązywał za pomocą programów, które Ci w tym celu udostępniłem do Twojej zabawy w Internecie).

Otoczenie zadanie to może wymagać na przykład tego, aby jako daną wejściową podać sieci informację, z jakiej części świata to zwierzę pochodzi. No bo jeśli z podanych danych wejściowych wynika, że zwierzę jest duże, ma rogi i daje mleko, to wiadomo, że jest to krowa. Ale o tym, czy może ona stanowić zagrożenie dla człowieka, decyduje kontynent, na którym to zwierzę spotkamy. Krowy europejskie i azjatyckie są bowiem z reguły spokojne i łagodne, natomiast niektóre krowy amerykańskie, hodowane na otwartych pastwiskach, bywają niebezpieczne. Dlatego w celu rozstrzygnięcia o tym, czy zwierzę może być groźne, czy nie – potrzebna jest wartość zmiennej określającej *pochodzenie* zwierzęcia. I tu pojawia się trudność: wiadomo, jak się nazywają poszczególne kontynenty, ale jak wprowadzić na przykład nazwę „Azja” albo „Ameryka” na wejście sieci neuronowej??!

Rozwiązaniem postawionego problemu jest użycie reprezentacji nazywanej „*jeden z N*”, gdzie N oznacza liczbę różnych możliwych wartości (nazw),

jakie może przyjmować zmienna nominalna. Sposób kodowania według metody „jeden z N” dla prostego przykładu, w którym  $N = 3$ , przedstawiłem na rysunku 2.29. Zasada jest prosta i polega na tym, że dla każdej zmiennej nominalnej, którą chcemy odwzorować na wejściu sieci, umieszcza się w warstwie wejściowej tej sieci tyle neuronów, ile różnych nazw może przybierać ta zmienna (czyli właśnie N). Jeśli na przykład założymy w rozważanym przykładzie, że rozpoznawane zwierzęta mogą pochodzić wyłącznie z Azji, Ameryki lub Europy, to dla reprezentowania na wejściu sieci zmiennej *Pochodzenie* musimy przeznaczyć zestaw trzech neuronów. Jeśli tak zrobimy i potem chcemy powiadomić sieć, że w danym momencie wartością zmiennej *Pochodzenie* jest nazwa *Ameryka*, to podajemy sygnał o wartości 0 na pierwsze z trzech zgrupowanych razem wejścia, wartość 1 na drugie z nich oraz ponownie wartość 0 na trzecie wejście.

Zmienna „*Pochodzenie*” może przyjmować następujące wartości  
**{Azja, Ameryka, Europa}**  
 Stosujemy następujące kodowanie  
 Azja: {1, 0, 0}  
 Ameryka: {0, 1, 0}  
 Europa: {0, 0, 1}



Rys. 2.29. Sposób kodowania zmiennej nominalnej na wejściu sieci neuronowej

Sądzę, że skoro czytasz tę książkę, to jesteś człowiekiem bystрыm i pomysłowym, więc na pewno czytając o metodzie „jeden z N” myślałeś sobie w duchu z poczuciem wyższości:

*Po co to tak komplikować? Ja mam lepszy pomysł! Zakodujmy sobie, że Azja to 1, Ameryka to 2, a Europa to 3, i wprowadzajmy te wartości na jedno wejście sieci! Albo uwzględniając, że sygnał na wejściu sieci powinien przyjmować wartości z przedziału od 0 do 1, może być tak: Azja to 0, Ameryka to ½, a Europa to 1. Dlaczego nikt wcześniej na to nie wpadł?*

Otoż to zaproponowane (hipotetycznie) przez Ciebie rozwiązanie nie jest niestety dobre.

Sieci neuronowe są bardzo wrażliwe na **wzajemne relacje** przedstawianych im wartości. Jeśli przyjmiesz pierwszą regułę kodowania, to sieć neuronowa podczas nauki będzie usiłowała wykorzystać w jakiś sposób fakt (wydedukowany z podawanych jej w zbiorze uczącym danych), że *Europa* to trzy

razy więcej niż *Azja* – i oczywiście wyjdą z tego same bzdury. Jeszcze gorzej będzie w przypadku danych mających charakter przeskalowany, bo wtedy będzie się wydawało, że *Amerykę* da się przeliczyć na *Europę* (mnożąc ją przez 2), ale *Azji* nie da się tak przeskalać, bo zero mnożone przez cokolwiek zawsze pozostaje zerem.

Słowem **musisz** się zgodzić na to, że zmienne nominalne trzeba koniecznie reprezentować techniką „jeden z N”, chociaż niestety zwiększa to liczbę wejść w sieci oraz powoduje zwiększenie liczby połączeń między warstwą wejściową a dalszymi warstwami sieci. Zwłaszcza ten drugi fakt jest kłopotliwy, bo musisz pamiętać o tym, że z każdym połączeniem w sieci neuronowej związany jest współczynnik wagowy, którego wartość trzeba potem pracowicie wyznaczyć w trakcie uczenia, więc dodatkowe wejścia (i dodatkowe połączenia) – to dodatkowe kłopoty przy uczeniu. Jednak mimo tych kłopotów nie ma innego (lepszego) rozwiązania i **musisz** (powtórzę to jeszcze raz) z wielokrotnią wejścia związane z każdą zmienną nominalną, stosując schemat „jeden z N” – no chyba że wymyślisz jakąś naprawdę skutecną i lepszą metodę ich kodowania.

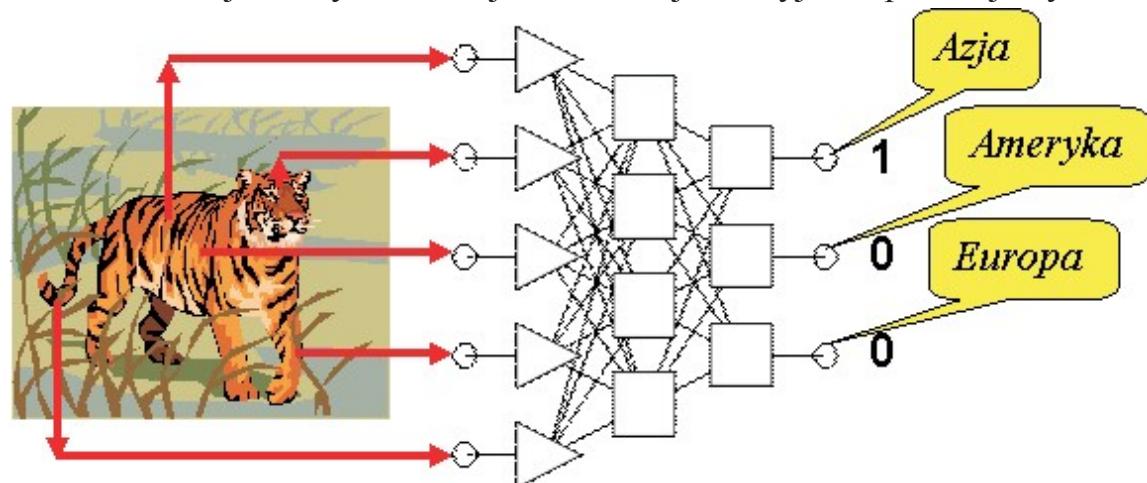
Zanim zakończę ten wątek, dodam tylko jeszcze jedną informację. Otóż zwykle tak jest, że od określonej reguły są także wyjątki. W przypadku reguły „jeden z N” wyjątkiem jest przypadek, kiedy N wynosi dwa. Dla takich „binarnych zmiennych nominalnych”, których przykładem może być zmienna *Płeć*, można wyjątkowo zastosować kodowanie polegające na tym, że przyjmujemy (przykładowo) następujące kody: *Mężczyzna* = 0 oraz *Kobieta* = 1 (lub odwrotnie) – i sieć sobie z tym poradzi. Ale takie binarne dane to w istocie typowy „wyjątek, który potwierdza regułę” – a regułą jest, że dane typu jakościowego **powinno się** kodować metodą „jeden z N”.

## 2.9. Jak interpretować odpowiedzi produkowane przez sieć?

Drugą wyróżnioną warstwą, którą omówimy w tym podrozdziale, jest **warstwa wyjściowa**, produkująca ostateczne rozwiązania postawionego problemu. Rozwiązania te są wysyłane na zewnątrz jako **sygnały wyjściowe** z całej sieci, więc ich interpretacja też jest dla nas ważna, bo musimy wiedzieć i rozumieć, o czym nas ta sieć chce powiadomić. Co do ilości neuronów wyjściowych to tu sytuacja jest prostsza niż w przypadku sygnałów wejściowych, bo na ogół wiemy, ile i jakich rozwiązań potrzebujemy i nie mamy tu rozterek typu – dodać dany sygnał czy też go nie dodawać, co zaprzątało naszą uwagę przy projektowaniu warstwy wejściowej sieci. Co do sposobów kodowania to sytuacja na wyjściu sieci jest podobna, jak na jej wejściu, to znaczy zmienne przyjmujące wartości liczbowe trzeba przeskalać tak, żeby wyjściowe neu-

rony mogły wyprodukować tę wartość, która jest prawidłowym rozwiązaniem postawionego problemu, a zmienne nominalne trzeba prezentować na wyjściu korzystając z metody „jeden z N”. Jednak na wyjściu sieci można się spodziewać paru problemów specyficznych właśnie dla tej warstwy, dlatego o kilku sprawach warto także i tu podyskutować.

Pierwszy problem szczegółowo związany jest ze stosowaniem metody „jeden z N”. Pamiętasz, że przy tej metodzie sygnał o maksymalnej wartości (wynoszącej zwykle 1) może mieć na swoim wyjściu tylko jeden neuron – ten przypisany do właściwej nazwy, będącej wartością zmiennej nominalnej. Wszystkie pozostałe neurony z grupy reprezentującej jedną zmienną powinny mieć wartości równe zero. Tę idealną sytuację wyraźnego wskazania przez sieć określonej nazwy zmiennej nominalnej na wyjściu pokazuje rysunek

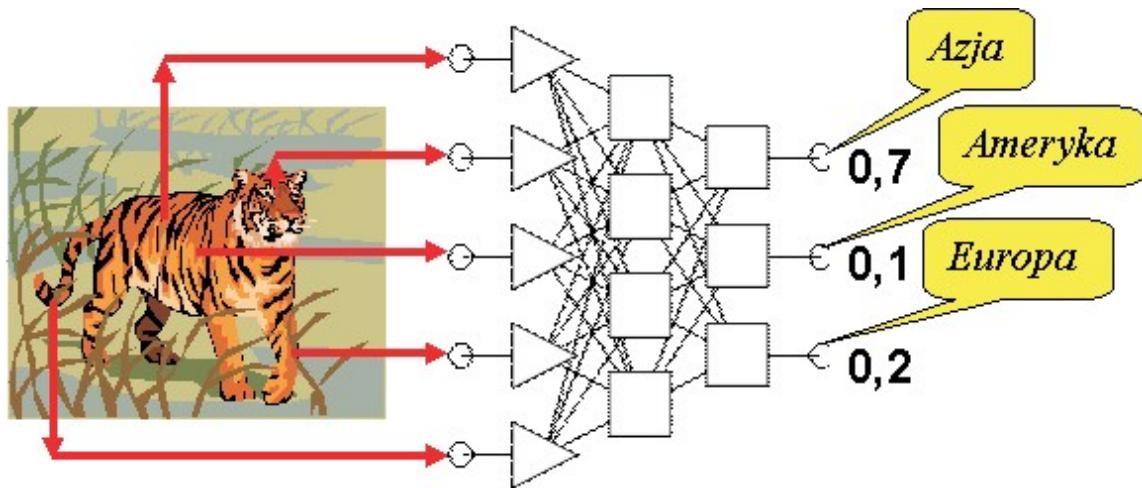


Rys. 2.30. Idealna sytuacja związana z użyciem zmiennej nominalnej na wyjściu sieci neuronowej

2.30. Na rysunku tym przyjęto konwencję, że zadanie przykładowo rozważanej sieci jest (w pewnym sensie) odwróceniem zadania z rysunku 2.29 – tam trzeba było podać **na wejściu**, na jakim kontynencie żyje klasyfikowane zwierzę (żeby dokonać klasyfikacji, czy jest ono niebezpieczne, czy nie), a tutaj **na wejściu** pokazywane<sup>1</sup> jest zwierzę, a sieć ma podać, na jakim kontynencie ono żyje.

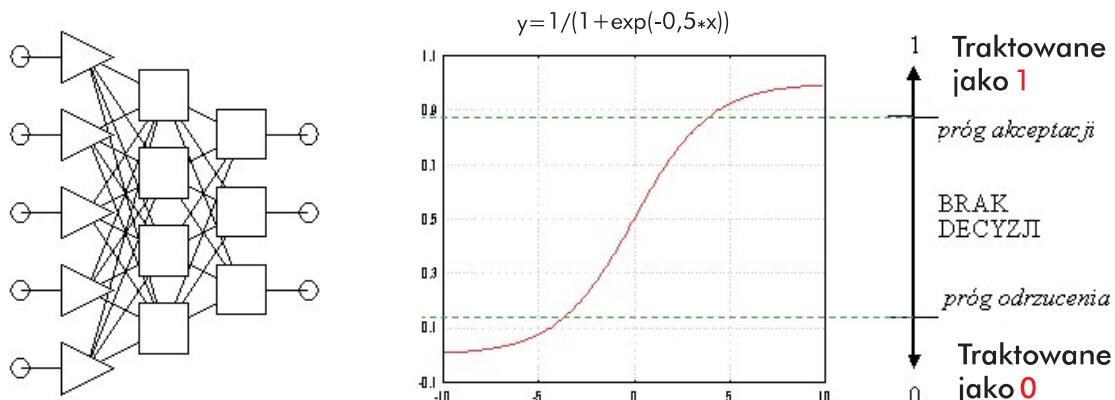
Taka sytuacja jak na rysunku 2.30 zdarza się jednak tylko w teorii (a w praktyce bywa spotykana jedynie jako wynik pomyślnego zbiegu okoliczności). W praktyce ze względu na ograniczoną dokładność pracy sieci, o czym

<sup>1</sup> „Pokazanie” sieci zwierzęcia polega na tym, że do neuronów wejściowej warstwy podawane są wybrane informacje o jego wyglądzie i budowie ciała (jaki ma kształt głowy, jaki kolor futra, jakie łapy, jaki ogon itp.). Symbolizują to czerwone strzałki na rysunku, łączące wybrane elementy budowy zwierzęcia z neuronami wejściowymi, do których dostarczana jest informacja na ich temat.



Rys. 2.31. Rzeczywisty rozkład wartości sygnałów w sieci neuronowej ze zmienną nominalną na wyjściu

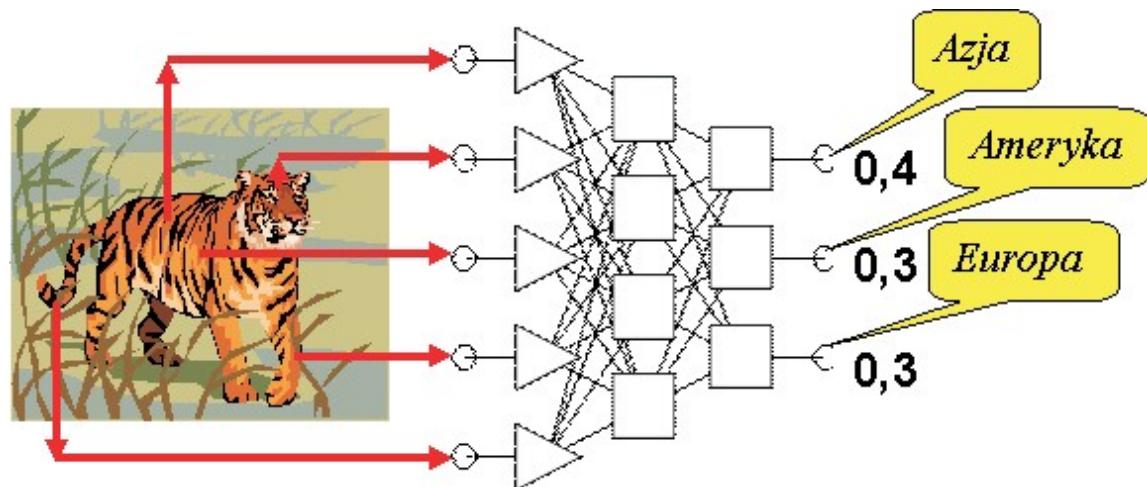
będzie jeszcze niżej mowa, prawie zawsze na wyjściach **wszystkich** neuronów, tworzących grupę przypisaną do jakiejś wyjściowej zmiennej nominalnej, pojawiają się jakieś niezerowe sygnały. Taka sytuacja pokazana jest na rysunku 2.31. No i co z tym zrobić?



Rys. 2.32. Efekt przetwarzania wykańczającego, które powoduje, że wartości wyjściowej zmiennej nominalnej mogą być wyznaczone jednoznacznie, pomimo niedokładnych wartości na wyjściach neuronów

Otoż do uzyskania zdecydowanych odpowiedzi w takich sytuacjach służy przyjęcie dodatkowych kryteriów obróbki wyników dostarczanych przez sieć neuronową w postaci *progu akceptacji* oraz *progu odrzucenia*. Istota tej koncepcji sprowadza się do tak zwanego **przetwarzania wykańczającego** (po angielsku *post-processing*). Sygnały wyjściowe obliczone przez końcowe neurony sieci są zgodnie z tą koncepcją kwantyfikowane poprzez porównanie ich aktualnych wartości ze wspomnianymi progami (patrz rysunek 2.32).

Wartości parametrów, jakimi są próg akceptacji i próg odrzucenia, mogą być dobierane zależnie od potrzeb, podobnie jak reguły rozstrzygania, co należy zrobić, jeśli na jednym lub kilku neuronach pojawi się sygnał braku decyzji. Doświadczenie uczy, że powinno się stawiać sieci **wysokie** wymagania, to znaczy nie próbować „na siłę” doprowadzić do jednoznacznej wartości zmiennej wyjściowej typu nominalnego, na przykład w tak słabo zdeterminowanej sytuacji, jak pokazana przykładowo na rysunku 2.33.



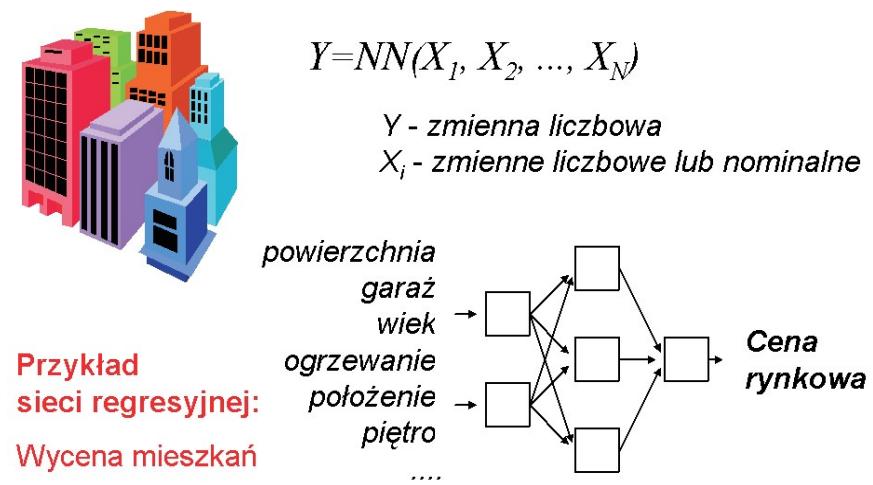
Rys. 2.33. Przykład rozkładu sygnałów wyjściowych, przy którym lepiej jest nie wyznaczać wcale wartości zmiennej nominalnej, niż narazić się na błąd, który jest w tej sytuacji bardzo prawdopodobny

Lepiej zawsze przyznać się do tego, że sieć nie potrafi dokonać jednoznacznej klasyfikacji wejściowego sygnału, niż podejmować konkretną decyzję w warunkach, kiedy z dużym prawdopodobieństwem może to być decyzja błędna.

## 2.10. Co lepiej starać się dostać z sieci – liczbę czy decyzję?

Korzystając z sieci neuronowych pamiętaj zawsze, że wyniki dostarczane przez sieć, nawet jeśli są wartościami liczbowymi (podlegającymi stosownemu skalowaniu), to mają zawsze charakter **przybliżony**. Jakość tego przybliżenia może być różna, jednak o dokładności wielu cyfr znaczących nie może tu być mowy – dobrze, jeśli wynik dostarczany przez neuron ma dokładność lepszą niż dwie cyfry (czyli błąd może dochodzić do kilku procent). Taka jest już po prostu natura tego narzędzia. Świadomość występowania tych ograniczeń zmusza do odpowiedniej **interpretacji** sygnałów wyjściowych, by można było z nich sensownie korzystać, a także skłania do zastanowienia nad tym, jaki model obliczeń neuronowych chcesz wykorzystać. Generalnie

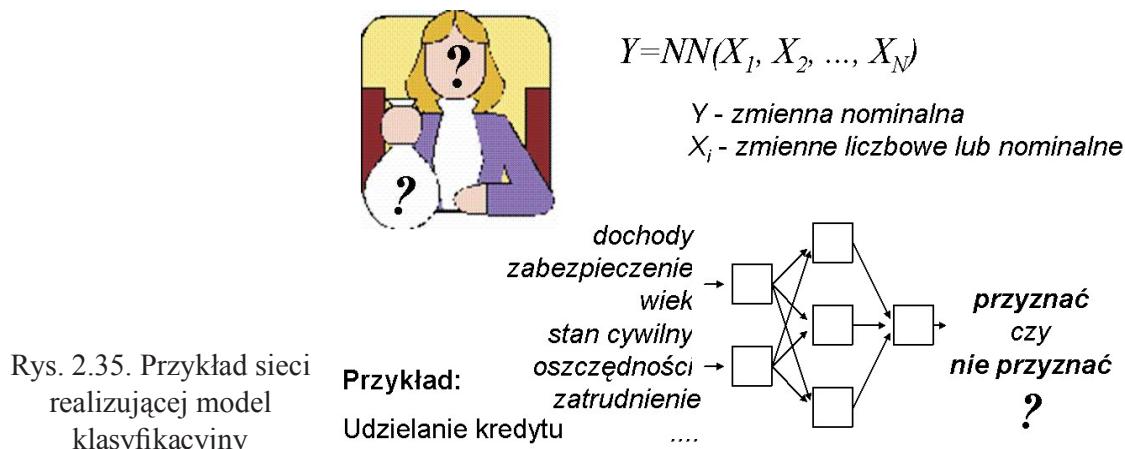
można powiedzieć, że sieci neuronowe mogą tworzyć modele dwóch typów: **regresyjne** oraz **klasyfikacyjne**. Model regresyjny to taki, w którym na wyjściu sieci oczekujemy (i wymagamy) podania konkretnej wartości liczbowej, będącej rozwiązaniem postawionego problemu. Interpretację takiego modelu ilustruje rysunek 2.34.



Rys. 2.34. Przykład sieci realizującej model regresyjny

W modelu podanym na rysunku zadaniem sieci neuronowej jest podanie oszacowania ceny mieszkania. Na wejściu podawane są dane, które mogą mieć charakter liczbowy (na przykład powierzchnia w  $m^2$ ) oraz dane, które mają charakter nominalny (na przykład to, czy z mieszkaniem związany jest garaż), natomiast na wyjściu oczekujemy wartości liczbowej, która określa kwotę, jaka zapewne będzie uzyskana przy sprzedaży mieszkania. Jak wiedzą wszyscy sprzedający lub kupujący mieszkania – ich cena rynkowa jest uzależniona od wielu czynników, przy czym nikt nie potrafi podać ścisłych reguł ekonomicznych, pozwalających z góry przewidzieć, że to mieszkanie będzie kosztowało tyle, a tamto na przykład dwa razy więcej. Pozornie jest to niemożliwe do przewidzenia, bo cena stanowi każdorazowo wynik wolnej gry rynkowej i suwerennych decyzji podejmowanych przez osoby sprzedające lub kupujące konkretne mieszkanie. A jednak okazuje się, że sieć neuronowa po odpowiednio długiej nauce (na podstawie danych dotyczących wcześniej zawieranych transakcji kupna-sprzedaży mieszkań) potrafi wytworzyć na tyle dobry model regresyjny tego problemu, że rzeczywiste ceny uzykiwane w kolejnych transakcjach różnią się od prognozy „odgadniętej” przez sieć zaledwie o kilka procent.

Alternatywny model (klasyfikacyjny) związany jest z wymaganiem użycia od sieci informacji o zaliczeniu obiektu opisanego przez dane wejściowe do jednej z możliwych klas. Oczywiście, przy tego typu zadaniu na wyjściu sieci umieszczona zostaje zmienna nominalna (patrz rys. 2.35).

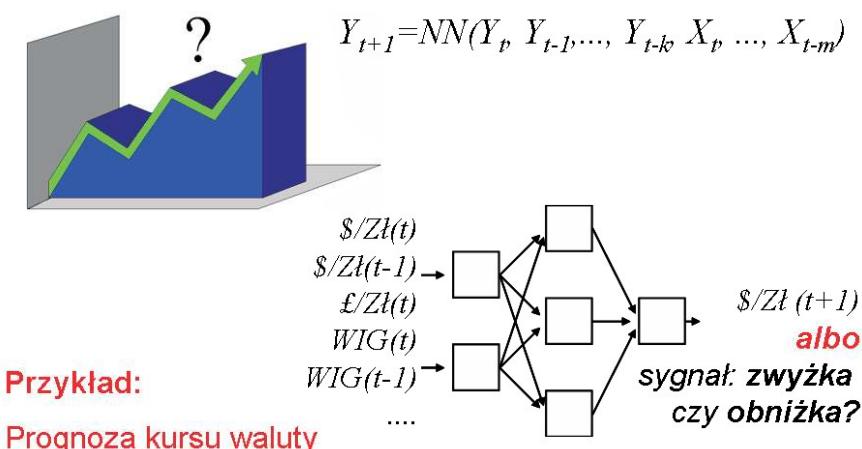


Pokazany na rysunku przykład dotyczy zadania klasyfikacji osób (albo firm) ubiegających się w banku o kredyt. Takiego klienta urzędnik bankowy musi zaliczyć do jednej z dwóch kategorii: Do pierwszej zalicza się klientów wiarygodnych i odpowiedzialnych, którym warto udzielić kredytu, bo spłacą go na pewno wraz z odsetkami przynosząc w ten sposób bankowi zysk. Do drugiej należą naciągacze i potencjalni bankruci, którzy kredytu nie spłacą, narażając bank na straty. Jak odróżnić jednych od drugich? Ścisłych reguł ani algorytmów nie ma, ale prawidłową decyzję może podpowiedzieć sieć neuronowa, nauczona na podstawie danych z przeszłości. Danych takich jest zwykle sporo, bo każdy bank wcześniej udzielał już wielu kredytów i ma pełne dane o swoich klientach, zarówno tych, którzy spłacili zaciągniętą pożyczkę, jak i tych, którzy dopuścili się defraudacji.

Doświadczenia wielu lat eksplotacji sieci neuronowych dowiodły, że najwygodniej jest tak interpretować stawiane neurokomputerowi zadania, by odpowiedź mogła być udzielona przy zastosowaniu modelu klasyfikacyjnego. Na przykład można domagać się, by sieć określiła, czy zyskowność inwestycji jest „mała”, „średnia” lub „duża”, względnie czy kredytobiorca jest „pewny”, „ryzykowny” lub „zupełnie niewiarygodny”. Natomiast wymaganie dokładnego określenia wielkości spodziewanego zysku, dokładności stopnia ryzyka lub wysokości kwoty, jaką można komuś pożyczyć, prowadzi będzie niezawodnie do frustracji, bo sieć na ogół tego zrobić nie potrafi.

**Dlatego liczba wyjść z budowanej sieci bywa często większa niż liczba pytań, na które poszukujemy odpowiedzi.** Dzieje się tak, ponieważ dla wielu sygnałów wyjściowych trzeba sztucznie wprowadzić kilka neuronów obsługujących dane wyjście – na przykład w ten sposób, żeby przewidywany przedział wartości sygnału wyjściowego został podzielony na pewne podzakresy, których rozróżnienie jest dla użytkownika istotne. W takim przypadku nie próbujemy sieci zmuszać do tego, żeby na swoim wyjściu podała konkretną „odgadywaną” wartość, natomiast działamy w taki sposób, że poszczególne

głórnne neurony wyjściowe odpowiedzialne są za **sygnalizowanie przynależności aktualnego rozwiązania do określonego przedziału** i to nam na ogół wystarczy. Taką sieć klasyfikacyjną znacznie łatwiej zbudować i nauczyć, podczas gdy tworzenie sieci, z której „wyciska się” dokładne rozwiązania problemów matematycznych, jest typową „sztuką dla sztuki” – czasochlonną zabawą o minimalnej przydatności praktycznej. Sformułowane tu wnioski podsumowuje rysunek 2.36, na którym pokazane jest jedno z „klasycznych” zadań, stawianych sieci neuronowej: prognozowanie kursów walut.



Rys. 2.36. Zadanie prognozowania kursów walut jest jednym z wielu zadań, w których twórca sieci neuronowej może wybierać pomiędzy modelem regresyjnym a modelem klasyfikacyjnym

Jak pokazano na rysunku, zadanie to można rozwiązać na dwa sposoby: Albo można budować model prognostyczny, który spróbuje przewidzieć, ile rubli będzie wart dolar w dniu jutrzejszym, albo można się zadowolić modelem, który będzie tylko sygnalizował, czy na drugi dzień należy oczekiwać podwyżki kursu, czy jego obniżenia? To drugie zadanie sieć rozwiąże znacznie łatwiej – a prognoza zwyczki lub obniżki kursu może być użyteczna dla kogoś, kto zamierza na przykład kupić dolary na wyjazd zagraniczny.

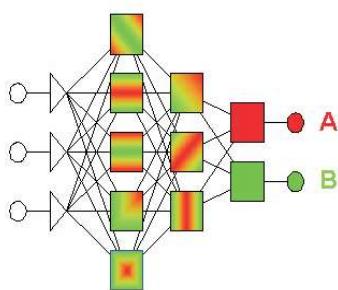
### 2.11. Czy lepiej mieć jedną sieć z wieloma wyjściami, czy kilka sieci o jednym wyjściu?

Z warstwą wyjściową sieci neuronowej wiąże się jeszcze jedno zagadnienie, które można rozwiązać na dwa sposoby, a wybór jednego z tych sposobów jest przedmiotem swobodnej decyzji twórcy sieci. Otóż, jak wiadomo, zawsze możliwe jest zbudowanie sieci, która będzie miała dowolną liczbę wyjść – tyle, ile danych wyjściowych chcielibyśmy uzyskać w wyniku rozwiązania postawionego problemu. Nie zawsze jednak jest to optymalne rozwiązanie,

ponieważ proces uczenia sieci o wielu wyjściach musi prowadzić – podczas ustalania wartości wag wewnątrz sieci – do swoistych kompromisów, które zawsze pogarszają wynik.

Kompromisy, o których była mowa, polegają przykładowo na tym, że przy ustalaniu zadań dla pewnego neuronu warstwy ukrytej trzeba uwzględnić (co wyraża się w ustaleniu wartości znajdujących się w nim współczynników wag), jaką rolę będzie ten neuron pełnił w obliczaniu wartości **kilku** neuronów wyjściowych, do których wysyła on swój sygnał wyjściowy, gdy już go obliczy. Może się więc zdarzyć, że rola danego neuronu ukrytego, optymalna z punktu widzenia obliczania z jego pomocą jakiegoś jednego wybranego sygnału wyjściowego z całej sieci, będzie istotnie odmienna niż jego rola optymalna dla któregoś innego wyjścia. W takim przypadku proces uczenia będzie co chwilę zmieniał wartości wag w tym neuronie ukrytym, dostosowując go raz do jednej, a raz do drugiej roli – z oczywistym skutkiem w postaci długiego i mało skutecznego uczenia. Dlatego często lepiej jest podzielić złożony problem i zamiast jednej sieci o wielu wyjściach zbudować kilka oddzielnych sieci, które wykorzystują ten sam zestaw danych wejściowych, ale mają rozdzielone warstwy ukryte oraz pojedyncze wyjście (rys. 2.37).

Załóżmy, że uczymy jedną sieć neuronową o dwóch wyjściach **A** oraz **B**.

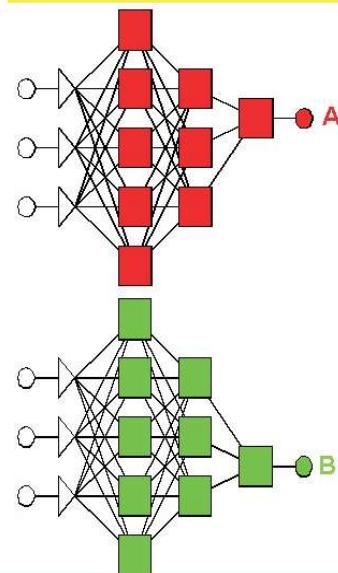


W neuronach obu warstw ukrytych będą musiały być zgromadzone informacje potrzebne do wyznaczania wartości **A** oraz **B**.

Czasem może to być korzystne, wtedy gdy między wyjściami zachodzi synergia i doskonalać pracę sieci zmierzającą do wyznaczania poprawnych wartości **A** przy okazji gromadzi się wiedzę przydatną przy wyznaczaniu 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:



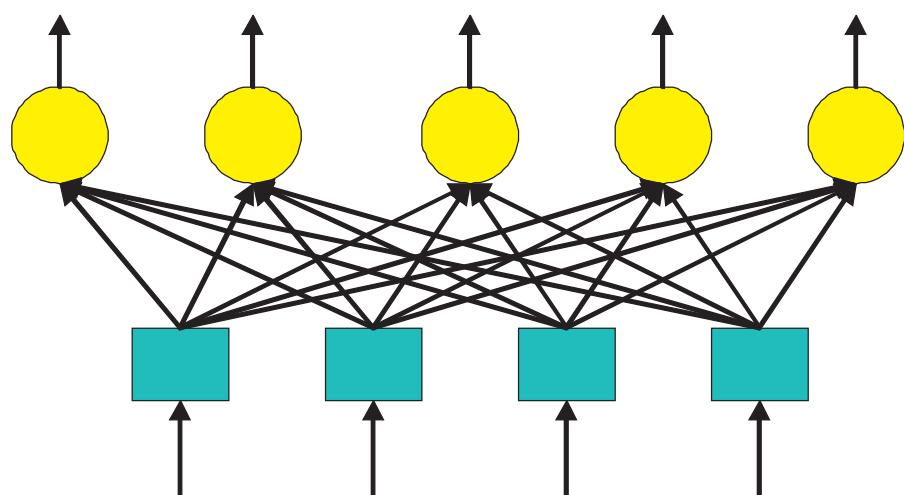
Każdą z nich można wtedy będzie w całości optymalnie doroślić do obliczania wymaganego wyjścia **A** albo **B**

Rys. 2.37. W jednej sieci o wielu wyjściach neurony ukryte muszą tak się uczyć, żeby obsłużyć obydwa wyjścia. Czasem zamiast stosować jedną sieć o wielu wyjściach, lepiej jest zastosować kilka oddzielnych sieci mających te same sygnały wejściowe, ale każdorazowo tylko jedno wyjście, bo neurony ukryte mogą się specjalizować

Reguły zasugerowanej w związku rysunkiem 2.37 nie należy jednak traktować zbyt dogmatycznie, gdyż czasem zdarza się, że sieć o wielu wyjściach uczy się lepiej od tej o jednym wyjściu. Ten paradoksalny na pozór wynik można uzasadnić faktem, że opisany w poprzednim akapicie potencjalny „konflikt” związany z funkcjonowaniem neuronów ukrytych i z ustalaniem ich roli podczas wypracowywania kilku różnych wartości wyjściowych może wcale nie wystąpić. Przeciwnie, niekiedy obserwuje się przy uczeniu sieci efekt swoistej synergii, polegający na tym, że ustalając (w toku procesu uczenia) parametry neuronu ukrytego, optymalne z punktu widzenia „interesów” kilku neuronów warstwy wyjściowej, osiąga się sukces szybciej i skuteczniej, niż gdy się to robi w oddzielnych sieciach, indywidualnie dostrajanych dla każdego sygnału wyjściowego osobno. Dlatego nie trzeba się z góry nastawiać na to, że z pewnością lepsze jest jedno albo drugie rozwiązanie, tylko korzystne jest wypróbowanie obydwu i wybranie lepszego z nich. Moja własna obserwacja, wynikająca z tego, że sam zbudowałem już setki sieci do różnych zastosowań, a także konsultowałem dziesiątki prac moich studentów i doktorantów, wskazuje na to, że **znaczco częściej** optymalnym rozwiązaniem jest jednak kolekcja sieci o pojedynczych wyjściach – chociaż biologiczne sieci neuronowe są częściej zorganizowane na zasadzie agregatów wielowyjściowych.

Wiesz już, że każda sieć *feedforward* musi mieć przynajmniej dwie wymienione wyżej warstwy – wejściową i wyjściową. Bywają zresztą takie sieci (rys. 2.38) – nazywa się je wtedy sieciami **jednowarstwowymi**, bo mają tylko jedną warstwę uczącą się. Jest to oczywiście warstwa wyjściowa, bo warstwa wejściowa, jak już wiesz, w żadnej sieci nie podlega uczeniu.

Jednak wiele sieci (zwłaszcza tych rozwiązywających bardziej złożone zadania) musi dysponować dodatkowymi warstwami elementów, pośredniczących



Rys. 2.38. Przykład sieci jednowarstwowej

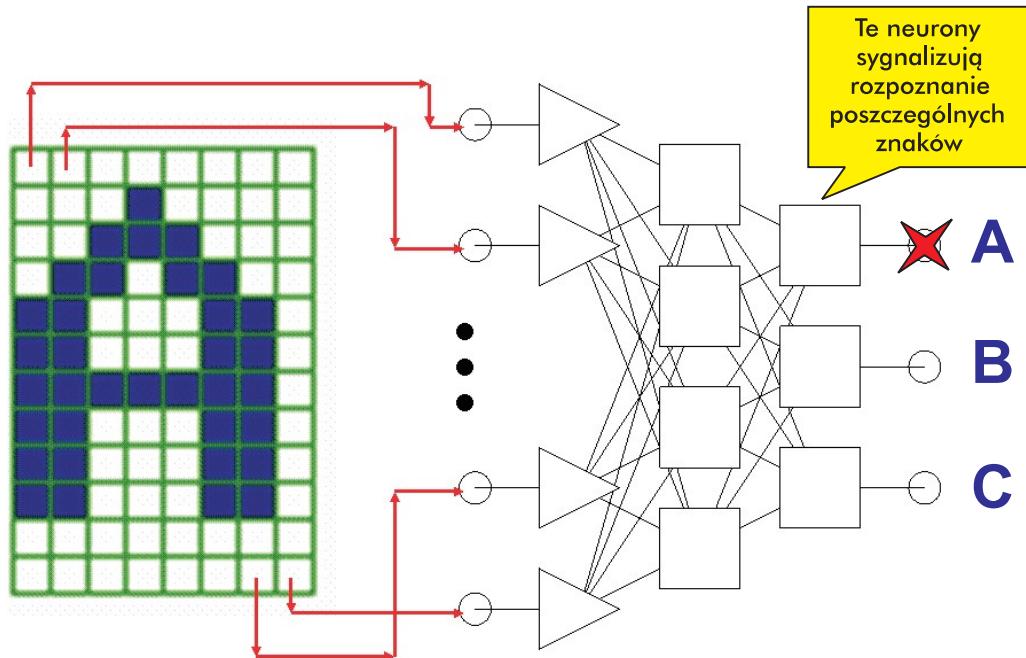
pomiędzy wejściem i wyjściem. Te warstwy wprowadzone pomiędzy wejście i wyjście nazywane są zwykle **warstwami ukrytymi**. Nazwa ta brzmi niezwykle i tajemniczo, dlatego możesz się czuć zakłopotany, stykając się z problemem warstw ukrytych po raz pierwszy. Chciałbym Ci w związku z tym jak najszybciej wyjaśnić, czym są te warstwy i w jakim sensie są one „ukryte”.

## 2.12. Co się kryje w warstwach ukrytych?

Najkrócej można powiedzieć, że warstwy ukryte stanowią narzędzie, służące do takiego przetworzenia sygnałów wejściowych (odebranych przez warstwę wejściową), by warstwa wyjściowa mogła łatwiej znaleźć potrzebną odpowiedź (rozwiązań problemu). Działanie tych dodatkowych (nie zawsze występujących) warstw nie jest możliwe do bezpośredniej obserwacji przez użytkownika, który stawia sieci zadania i obserwuje, czy sieć poprawnie je wykonuje. Nie ma on dostępu ani do wejść neuronów tych warstw (przy przesyłaniu do nich sygnałów trzeba korzystać z pośrednictwa neuronów warstwy wejściowej), ani do ich wyjść (efekty działania neuronów warstw ukrytych ujawniają się wyłącznie pośrednio, poprzez odpowiedzi, jakie produkują i wysyłają neurony warstwy wyjściowej). Jak z tego wynika – działanie neuronów warstw pośredniczących nie jest bezpośrednio widoczne dla użytkownika sieci – i w tym właśnie sensie są one *ukryte*.

Chociaż ukryte – te dodatkowe warstwy w sieciach neuronowych pełnią bardzo ważną i odpowiedzialną funkcję. Tworzą one dodatkową strukturę przetwarzającą informację, a ich rolę najłatwiej jest przedyskutować na przykładzie sieci realizujących często pojawiające się zadanie **rozpoznawania obrazów**. W sieciach takich na wejście (do pierwszej warstwy sieci) podaje się obraz cyfrowy. Nie może to być obraz z typowego aparatu fotograficznego ani zwykły tradycyjny obraz wczytany do systemu przez skaner lub *Frame Grabber*. Powód jest bardzo prosty: w takim systemie zasilanym na wejściu samym obrazem, jakim takim, a nie jakimś jego bardziej wyrafinowanym opisem (porównaj rysunek 2.30) wejściowa warstwa neuronów odpowiada swymi rozmiarami i organizacją (zgrupowaniem neuronów w odpowiednie wiersze i kolumny) organizacji samego obrazu. Po prostu do każdego punktu obrazu przypisany jest neuron wejściowy sieci, który analizuje i sygnalizuje jego stan. Obraz z cyfrowego aparatu fotograficznego lub ze skanera ma postać zbioru pikseli, ale tych pikseli są **miliony**. Niepodobna zbudować sieć, która by na wejściu miała miliony neuronów wejściowych, zwłaszcza że taka sieć musiałaby mieć miliardy (!) połączeń, dla których trzeba byłoby wyznaczać wartości wag. Taka konstrukcja jest całkowicie niemożliwa do realizacji.

Można jednak sobie wyobrazić system, który na wejściu będzie otrzymał obraz cyfrowy bardzo uproszczony, złożony z niewielkiej liczby pikseli, układających się w formę nieskomplikowanych znaków – na przykład liter (rys. 2.39).



Rys. 2.39. Przykładowa sieć rozpoznająca proste obrazy

Na wyjściu takiej sieci oczekuje się decyzji informujących o tym, co rozpoznano – na przykład może być ona tak zorganizowana, że do poszczególnych neuronów wyjściowych przypiszemy umownie pewne decyzje – na przykład „rozpoznano literę A”, „rozpoznano literę B” itp., a wielkości sygnałów na tych wyjściach interpretować się będzie w kategoriach stopnia pewności odpowiedniej decyzji. Łatwo zauważyć, że możliwe jest w związku z tym podawanie przez sieć odpowiedzi wieloznacznych („to coś jest podobne w stopniu 0,7 do litery A, ale w stopniu 0,4 przypomina także literę B”). Jest to jedna z ciekawych i użytecznych cech sieci neuronowych, które można w związku z tym kojarzyć z systemami o rozmytej (fuzzy) logice działania – obszerniejsze omówienie tego aspektu wykracza jednak poza tę książkę.

Neurony warstwy ukrytej pełnią w omawianej tu sieci rolę **pośredników** – mają one bezpośredni dostęp do danych wejściowych, czyli oglądają pokazany sieci obraz, a na podstawie ich wyjść dalsze warstwy podejmują określone decyzje o rozpoznaniu takiego lub innego obrazu. Dlatego uważa się, że rola neuronów warstwy ukrytej polega na tym, by wypracowywały zestawy takich **wstępnie przetworzonych** danych wejściowych, z których korzystać będą neurony warstwy wyjściowej przy określaniu końcowego wyniku.

Przydatność tych warstw pośrednich wynika z faktu, że na ogólny dokonanie pewnych przekształceń danych wejściowych sprawia, że rozwiązanie stawianego przed siecią zadania staje się znacznie łatwiejsze niż w przypadku próby rozwiązywania zadania w sposób bezpośredni. Na przykład, w rozważanym tu zadaniu rozpoznawania obrazów trudno jest znaleźć regułę pozwalającą ustalić, jaki obiekt pokazano, gdy analizuje się wyłącznie same jasne i ciemne piksele obrazu (a tylko taką informacją dysponuje sieć bez warstwy ukrytej). Znalezienie prawidłowej reguły rozpoznawania funkcjonującej na tak niskim poziomie jest trudne, ponieważ rozpoznawane obiekty mogą występować w wielu odmianach różniących się nieco kształtem, ale mających to samo znaczenie. Ten sam obiekt (na przykład litera A) może bowiem wyglądać trochę inaczej, gdy jest skanowany z rękopisu, inaczej gdy pochodzi z tekstu drukowanego, a jeszcze inaczej, gdy został na przykład sfotografowany na powiewającym transparencie. Nawet litera wydrukowana dokładnie tą samą czcionką może być reprezentowana przez dwa obrazy mające zupełnie inne wartości pikseli w pewnych wybranych punktach, jeśli jest na przykład przesunięty (takie banalne zakłócenie, w niczym nie przeszkadzające człowiekowi, który czyta tekst, może zmienić obraz cyfrowy litery w zupełnie dramatyczny sposób, jeśli musimy przywiązywać wagę do tego, który konkretnie piksel jest biały, a który czarny). Jakby tego było mało – przy obrazach cyfrowych o małej rozdzielczości pojawia się dodatkowa trudność polegająca na tym, że zupełnie różne obiekty mogą mieć na cyfrowych obrazach bardzo duże zbiory identycznych pikseli.

Oczekiwanie, że sieć neuronowa „jednym skokiem” pokona wszystkie te trudności, jakie dzielą „surowy” obraz od finalnej decyzji dotyczącej rozpoznania tego, co ten obraz przedstawia – byłoby w takim przypadku mało realistyczne. Rozumowanie to jest prawidłowe, gdyż odpowiednie doświadczenie potwierdza, że żaden proces uczenia nie zdoła zmusić prostej sieci bez warstwy ukrytej do tego, by potrafiła to zadanie rozwiązać. Taka sieć bez warstw ukrytych musiałaby sobie wytworzyć jakiś taki zdumiewający mechanizm, dzięki któremu raz rozumiałaby pewne wybrane i ustalone zestawy pikseli jako należące do różnych obiektów, a innym razem kojarzyłaby różne zestawy pikseli z tym samym obiektem. Tego się po prostu zrobić nie da.

To, czego nie może zrobić sieć o mniejszej liczbie warstw, potrafi na ogólny wykonać sieć zawierającą warstwę ukrytą. Neurony tej warstwy będą w takich zadaniach znajdować pewne wartości pomocnicze, znaczco ułatwiające rozwiązanie postawionego zadania. Na przykład, w rozważanym zadaniu rozpoznawania obrazów neurony warstwy ukrytej mogą wykrywać i kodować ogólne **cechy** opisujące strukturę obrazu i widocznych na nim obiektów. Cechy te powinny lepiej odpowiadać wymaganiom związanym z ostatecznym

rozpoznawaniem obrazu niż sam oryginalny obraz – na przykład mogą być w pewnym stopniu niezależne od położenia czy skali rozpoznawanych obiektów. Przykładem cech opisujących obiekty na obrazach liter (ułatwiających ich rozpoznawanie) mogą być:

- obecność lub brak w rysunku litery zamkniętych konturów (posiadają je litery O, D, A i kilka innych, nie mają tej cechy litery I, T, S, N itd.),
- czy znak ma wcięcie u dołu (A, X, K) albo u góry (U, Y, V, K), a może z boku (E, F),
- czy ma kształt raczej zaokrąglony (O, S, C), czy ostry (E, W, T, A).

Oczywiście cech różnicujących litery i równocześnie niewrażliwych na typowe czynniki utrudniające ich rozpoznawanie (zmiana kroju czcionki lub pisma ręcznego, zmiana wielkości litery, jej pochylenie, przesunięcie itd.) można znaleźć znacznie więcej. Należy jednak z naciskiem podkreślić, że twórca sieci neuronowej nie musi sam jawnie zadawać, jakie to cechy obrazu mają być znajdowane, gdyż sieć może sama nabywać stosownych umiejętności w trakcie procesu uczenia. Oczywiście, nie ma żadnej gwarancji, że sieć neuronowa przeznaczona do rozpoznawania liter nauczy swoją warstwę ukrytą wykrywania i sygnalizowania właśnie tych cech, które wyżej wymieniłem. Można nawet się założyć o dużą sumę, że tych właśnie, przeze mnie wskazanych cech różnicujących litery, sieć neuronowa sama nie wykryje. Jednak doświadczenia niezliczonych badaczy potwierdzają, że sieci neuronowe w zdaniach **rozpoznawania** potrafią znajdować cechy, które **skutecznie** ułatwiają rozpoznawanie – chociaż na ogół twórca sieci nie potrafi zrozumieć, jaki sens ma taka lub inna cecha kodowana w warstwie ukrytej.

Użycie warstwy ukrytej jest więc ze wszech miar celowe, bo sieć neuronowa może dzięki niej przejawiać znacznie inteligentniejsze formy zachowania, trzeba jej jednak dać szansę poprzez uwzględnienie w strukturze elementów, które mogą właśnie takie „opisowe cechy obrazu” wydobywać. Warto dodać, że charakter wydobywanych cech nie jest zdeterminowany przez strukturę sieci, tylko przez proces uczenia. Jeśli wyobrażymy sobie sieć, która ma rozpoznawać obrazy, to cechy, które będą wydobywały neurony warstwy ukrytej, dostosują się automatycznie do rodzaju rozpoznawanych obrazów. Jeśli nakażemy sieci wykrywanie na zdjęciach lotniczych obrazów zamaskowanych wyrzutni rakietowych, to staje się oczywiste, że głównym zadaniem warstwy ukrytej będzie uniezależnienie się od położenia obiektu, gdyż podejrzany kształt może się pojawić w dowolnym rejonie obrazu i zawsze powinien być tak samo rozpoznany. Jeśli natomiast sieć ma rozpoznawać litery, to nie powinna gubić informacji o ich położeniu (mało przydatna będzie sieć, która poinformuje nas, że gdzieś tam na zeskanowanej stronicy znajduje się litera A – my musimy wiedzieć, gdzie ona jest, a dokładniej – w jakim kontekście

występuje), więc w tym przypadku celem warstwy ukrytej może być wydobycie cech pozwalających na niezawodne rozpoznanie litery niezależnie od jej rozmiaru i niezależnie od kroju czcionki (font). Co ciekawe – oba te zadania może rozwiązywać (po odpowiednim treningu) ta sama sieć – chociaż oczywiście sieć nauczona rozpoznawać czołgi nie potrafi odczytywać pisma, a nauczona identyfikować odciski palców nie poradzi sobie z rozpoznawaniem twarzy.

### 2.13. Ile potrzeba neuronów, żeby otrzymać dobrze działającą sieć?

Jak wynika z przytoczonych uwag – najszerze możliwości zastosowań mają sieci posiadające przynajmniej trójwarstwową strukturę, z wyróżnioną warstwą wejściową przyjmującą sygnały, warstwą ukrytą wydobywającą potrzebne cechy wejściowych sygnałów oraz z warstwą wyjściową podejmującą ostateczne decyzje i podającą rozwiązanie. W tej strukturze pewne elementy są zdeterminowane: liczba elementów wejściowych i wyjściowych, a także zasada połączeń (każdy z każdym) pomiędzy kolejnymi warstwami. Są jednak elementy zmienne, które trzeba wybrać samemu: jest to **liczba warstw ukrytych** (jedna czy kilka?) oraz **liczba elementów w warstwie** (warstwach?) ukrytych (rys. 2.40).



Rys. 2.40. Najbardziej istotny problem przy wyborze struktury sieci neuronowej wiąże się z ustaleniem, ile elementów powinna mieć warstwa ukryta?

Ponieważ mimo wielu lat rozwoju tej techniki wciąż nie udało się stworzyć precyzyjnej teorii sieci neuronowych – elementy te wybiera się zwykle arbitralnie albo metodą prób i błędów. Zdarza się, że pomysł twórcy sieci na

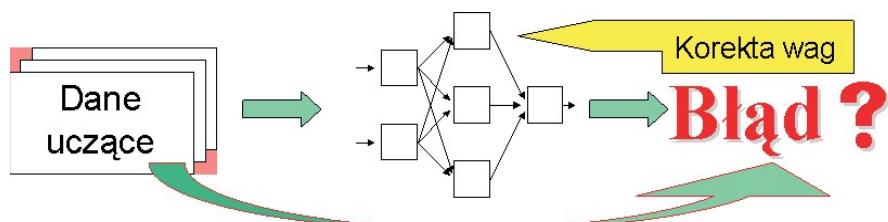
temat tego, ile neuronów ukrytych należy zastosować, a także jak je rozmieścić (np. w postaci jednej warstwy ukrytej albo w postaci kilku takich warstw), nie jest najbardziej trafny. Jednak nie powinno to mieć **zasadniczego** wpływu na sposób działania sieci, gdyż w trakcie procesu uczenia ma ona zawsze możliwość skorygowania ewentualnych błędów struktury poprzez wybór odpowiednich parametrów połączeń. Trzeba jednak wyraźnie przestrzec przed dwoma rodzajami błędów, stanowiących pułapki dla wielu (zwłaszcza początkujących) badaczy sieci neuronowych.

Pierwszy błąd polega na zaprojektowaniu sieci o zbyt małej liczbie elementów – jeśli warstwy ukrytej nie ma wcale lub gdy występuje w niej zbyt mało neuronów, proces uczenia może się definitywnie nie udać, gdyż sieć nie ma szans odwzorować w swojej (zbyt ubogiej) strukturze wszystkich szczegółów i niuansów rozwiązywanego zadania. Pokażę Ci potem konkretne przykłady ilustrujące fakt, że za mała i zbyt prymitywna sieć nie potrafi rozwiązać pewnych zadań – nawet jeśli się ją uczy bardzo długo i bardzo dokładnie. Po prostu z sieciami neuronowymi bywa tak jak z ludźmi: nie wszystkie sieci są wystarczająco uzdolnione, żeby rozwiązać określone zadanie. Na szczęście możesz zawsze bardzo łatwo poznać, czy sieć jest bardziej, czy mniej inteligentna, bo widzisz jej strukturę i możesz policzyć neurony, a miarą zdolności sieci jest tu po prostu liczba neuronów ukrytych. U ludzi trudniej to rozpoznać!

Niestety, mimo swobody budowania większych albo mniejszych sieci zdarza się czasem, że ktoś zaprojektuje sieć o zbyt małej inteligencji. Skutkuje to zawsze niepowodzeniem przy próbie zastosowania takiej sieci do jakiegoś konkretnego celu, bo taki „neuronowy głupek” mający zbyt mało neuronów ukrytych nigdy nie rozwiąże stawianych mu zadań, niezależnie od tego, jak bardzo będziesz się trudził, usiłując go czegoś nauczyć.

Niestety z inteligencją sieci nie można także „przedobrzyć”. Mówiąc obrazowo, efektem **nadmiernej** inteligencji sieci wcale nie jest większa sprawność w rozwiązywaniu stawianych jej zadań, tylko zdumiewający efekt polegający na tym, że sieć zamiast pracowicie zdobywać użyteczną wiedzę – zaczyna oszukiwać nauczyciela i w efekcie wcale się nie uczy! Brzmi to w pierwszej chwili wręcz nieprawdopodobnie, ale taka jest prawda. Sieć, która ma zbyt wiele warstw ukrytych lub zbyt wiele elementów w warstwach ukrytych, ma skłonność do upraszczania sobie zadania i w efekcie jeśli tylko może, „idzie na łatwiznę”. Żebyś zrozumiał, na czym to polega, muszę Ci w kilku słowach powiedzieć, jak przebiega proces uczenia sieci (rys. 2.41).

Szczegóły tego procesu poznasz w jednym z dalszych rozdziałów, ale już teraz muszę Ci powiedzieć, że sieć naucza się podając jej na wejście sygnały, dla których znane są prawidłowe rozwiązania, bo są one zawarte w **danych**



Rys. 2.41. Maksymalnie uproszczony schemat procesu uczenia sieci neuronowej

**uczących.** Dla każdego podanego zestawu danych wejściowych sieć usiłuje podać na wyjściu swoją propozycję rozwiązania. Na ogół rozwiązania wyznaczone przez sieć są inne niż prawidłowe rozwiązania podane w danych uczących, więc po skonfrontowaniu tego rozwiązania, które podaje sieć, z tym rozwiązaniem, które znajduje się w danych uczących jako **wzorzec poprawnego rozwiązania** – ujawniane jest, jak duży błąd sieć popełniła w swoim rozwiążaniu. Na podstawie oceny błędu algorytm nauczający sieć zmienia wagi wszystkich jej neuronów w taki sposób, żeby na przyszłość sieć już nie popełniała takich samych błędów.

Naszkicowany wyżej schemat procesu uczenia wskazuje, że sieć dąży do tego, **żeby nie popełniać błędów podczas prezentowania jej danych uczących**. Dobrze ucząca się sieć szuka w związku z tym takiej reguły przetwarzania sygnałów wejściowych, by w następstwie jej zastosowania nauczyć się zasad obliczania poprawnych rozwiązań. Jeśli sieć odkryje taką regułę, to potem potrafi rozwiązywać zarówno te zadania, które jej pokazywano w ramach danych uczących, jak i inne, podobne zadania, które będą jej przedstawiane w trakcie normalnej eksploatacji. Mówimy wtedy, że sieć wykazuje zdolność do uczenia się oraz do **generalizacji** (uogólniania) wyników uczenia i traktujemy to jako sukces.

Niestety, nadmiernie inteligentna sieć, mająca pokaźne zasoby pamięciowe w postaci dużej liczby neuronów ukrytych (wraz z ich nastawialnymi zestawami wag), może łatwo uniknąć popełniania błędów w trakcie procesu uczenia w taki sposób, że nauczy się całego zbioru danych uczących na pamięć. Wtedy niesłychanie szybko odnosi duże sukcesy w procesie uczenia, ponieważ na każde zadane jej pytanie zna i podaje prawidłową odpowiedź. Niestety, przy tym sposobie „fotografowania” danych uczących sieć ucząc się na podstawie przedstawianych jej przykładów poprawnych rozwiązań stawianych jej zadań nie podejmuje próby uogólniania nabywanych wiadomości, tylko usiłuje osiągać sukces na zasadzie dokładnego zapamiętania reguł w rodzaju „przy takim wejściu takie wyjście”.

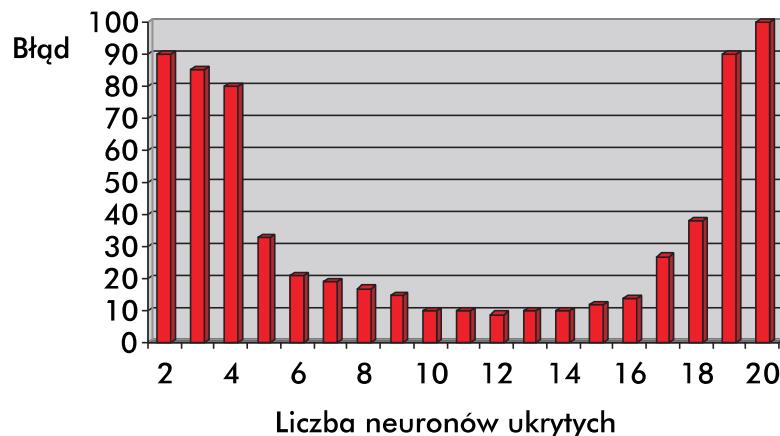
Objawem takiego nieprawidłowego działania sieci jest fakt, że uczy się ona dokładnie i szybko całego tzw. ciągu uczącego (czyli zbioru przykładów

użytych do pokazania sieci, jak należy rozwiązywać stawiane zadania), natomiast fatalnie kompromituje się przy pierwszej próbie **egzaminu**, czyli rozwiązyania zadania z podobnej klasy, ale jednak nieco odmiennego od zadań jawnie pokazywanych w trakcie uczenia. Przykładowo ucząc sieć rozpoznawania liter bardzo szybko uzyskujemy sukces (sieć bezbłędnie rozpoznaje wszystkie pokazywane jej litery), ale próba pokazania jej litery napisanej odmiennym charakterem pisma lub wydrukowanej innym fontem – prowadzi do zupełnego braku rozpoznania (na wszystkich wyjściach sieci są zera) lub do rozpoznań ewidentnie błędnych. Bliższa analiza wiedzy zgromadzonej przez sieć ujawnia w takich przypadkach, że zapamiętała ona wiele prostych reguł w rodzaju „jak tutaj są dwa piksele zapalone, a tam jest pięć zer – to należy rozpoznać literę A”. Te prymitywne reguły nie wytrzymują oczywiście konfrontacji z nowym zadaniem i sieć zawodzi nasze oczekiwania.

Opisany objaw „uczenia się na pamięć” nie występuje w sieciach mających mniejszą warstwę ukrytą, gdyż dysponując ograniczoną pamięcią sieć taka musi lepiej się starać i wypracować na niewielu dostępnych jej elementach warstwy ukrytej takie reguły przekształcania wejściowego sygnału, by umożliwić jego trafne wykorzystanie w więcej niż jednym przypadku wymaganej odpowiedzi systemu. Z reguły prowadzi to do znacznie wolniejszego i bardziej uciążliwego procesu uczenia (przykłady, na podstawie których sieć ma się uczyć, muszą być pokazane więcej razy – często kilkaset lub kilka tysięcy razy), jednak efekt końcowy jest zwykle znacznie lepszy. Po zakończeniu prawidłowo prowadzonego procesu uczenia z chwilą stwierdzenia dobrego działania sieci dla przykładów stanowiących podstawę uczenia mamy prawo przypuszczać, że poradzi sobie ona także z podobnymi (choć nie identycznymi) zadaniami, pokazanymi jej w trakcie egzaminu. Nie zawsze tak jest, ale zwykle tak **bywa** i na tym musimy opierać swoje oczekiwania dotyczące użycia sieci.

Zapamiętajmy więc na koniec następującą regułę: Nie należy oczekiwać cudów, dlatego przypuszczenie, że mało skomplikowana sieć, mająca niewiele ukrytych neuronów, rozwiąże skomplikowane zadanie, jest raczej przypuszczeniem mało realistycznym. Jednak zastosowanie zbyt wielu warstw ukrytych lub zbyt wielu neuronów prowadzi także do znacznego pogorszenia sprawności procesu uczenia. Gdzieś pomiędzy tymi skrajnościami jest zlokalizowana optymalna wielkość warstwy ukrytej.

Rysunek 2.42 pokazujący (na podstawie konkretnych symulacji komputerowych), jak wygląda błąd popełniany przez sieć przy różnej liczbie neuronów ukrytych, potwierdza, że jest wiele sieci, które działają prawie tak samo dobrze, a mają różną liczbę neuronów ukrytych – więc w to obszerne optimum nie tak trudno jest się „wstrzelać”. Jednak skrajności (to znaczy zbyt dużych i zbyt małych sieci) należą unikać. Szczególnie szkodliwe jest



Rys. 2.42. Przykładowa zależność błędu popełnianego przez sieć od liczby neuronów ukrytych

użycie dodatkowych (nadmiarowych) warstw ukrytych i nie należy się specjalnie dziwić, że często lepsze wyniki daje sieć o mniejszej liczbie warstw ukrytych (bo można ją porządnie nauczyć) niż – teoretycznie lepsza – sieć z większą liczbą warstw ukrytych, w której jednak proces uczenia „grzeźnie” w nadmiarze szczególnów. Dlatego należy stosować sieci o jednej lub (wyjątkowo!) dwóch warstwach ukrytych, natomiast pokusę stosowania sieci o większej liczbie warstw ukrytych najlepiej przewyścięgać stosując post i zimne kapiele.

## 2.14. Pytania kontrolne i zadania do samodzielnego wykonania

1. Wymień kilka najbardziej istotnych różnic, jakie występują pomiędzy sztucznymi sieciami neuronowymi a rzeczywistymi biologicznymi strukturami, dającymi się zidentyfikować w mózgu człowieka i zwierząt. Spróbuj odpowiedzieć, które z tych różnic są wynikiem **kapitulacji** (nie udaje się nam zbudować dostatecznie skomplikowanego sztucznego neuronu, więc zadowalamy się namiastką), a które są wynikiem świadomego, celowego wyboru twórców sieci neuronowych?

2. Omów pojęcie **wagi synaptycznej** od strony roli, jaką odgrywa w sztucznych sieciach neuronowych oraz od strony właściwości biologicznych, które opisuje. W układach elektronicznych modelujących sieci neuronowe wagi są niekiedy **dyskretne**, to znaczy mogą przyjmować wyłącznie niektóre wartości (na przykład całkowito-liczbowe) – a nie całkiem dowolne. Jak sądzisz, jaki to ma wpływ na działanie sieci?

3. Przedstaw schemat przetwarzania informacji wejściowych w sygnał wyjściowy, zachodzący w sztucznym neuronie oraz wskaż, czym mogą się

różnić od siebie poszczególne warianty sztucznych neuronów (liniowe, MLP, RBF).

4. Omów krok po kroku działanie sieci zbudowanej ze sztucznych neuronów. Dlaczego warstwa ukryta nosi taką właśnie nazwę?

5. Omów problem związku **struktury** sieci neuronowej i wykonywanych przez nią **funkcji**. Jakie bywają najczęściej spotykane struktury sieci i jakie są ich właściwości? Jak wyjaśnić możliwość wypełniania sensownych zadań obliczeniowych przez sieć, która ma losowe (przypadkowe) połączenia elementów?

6. Jak sądzisz, o czym świadczy fakt, że sieć neuronowa wykazuje dużą odporność na uszkodzenia jej elementów oraz całej struktury?

7. Co powoduje, że w sieciach neuronowych tak radykalnie trzeba odróżnić dane typu **ilościowego** od danych typu **jakościowego** (danych nominalnych)? Istnieje opinia, że sieć, w której wykorzystuje się na wejściu lub na wyjściu dane nominalne, wymaga dłuższego uczenia, ponieważ jest w niej więcej połączeń, których wagę trzeba ustalić. Czy jest to pogląd uzasadniony?

8. Sieć która ma na wyjściu dane kodowane metodą „*jeden z N*” może czasem odmawiać udzielenia odpowiedzi na postawione jej pytanie. Jaki może być skutek tego faktu? Czy wskazana okoliczność powinna być rozumiana jako zaleta, czy jako wada takiej sieci?

9. Czy zadanie analizy serii czasowych i związanej z tym prognozy (patrz rys. 2.36) można uznać za bardziej podobne do zadania regresyjnego (rys. 2.34), czy do zadania klasyfikacyjnego (rys. 2.35)? Jakie argumenty przemawiają za jedną, a jakie za drugą decyzją?

10. Czy z rozważań przedstawionych na rysunku 2.37 (oraz w tekście powiązanym z tym rysunkiem) należy wyciągać wniosek, że zawsze niekorzystne jest stosowanie sieci neuronowej o wielu wyjściach? W szczególności czy mając problem, w którym na wyjściu sieci oczekujemy rozwiązania w postaci zmiennej nominalnej, kodowanej techniką „*jeden z N*”, można zamiast jednej sieci o N wyjściach zastosować N sieci, z których każda będzie miała jedno wyjście?

11. **Zadanie dla zaawansowanych:** Wyobraźmy sobie, że w pewnym zadaniu klasyfikacyjnym, które chcemy rozwiązać z użyciem sieci neuronowych, mamy do dyspozycji pewną liczbę sygnałów wejściowych, co do których istnieją wątpliwości, czy wnoszą one jakieś wartościowe informacje w kontekście rozważanej klasyfikacji, czy też nie. Na przykład, czy wynik pewnej analizy krwi pozwala zaliczyć pacjenta do grupy osób dobrze znoszących proponowane leczenie, czy przeciwnie – ten wynik niczego w przedmiotowej sprawie nie wyjaśnia. Otóż mając takie wątpliwe dane wejściowe, możemy przezywać rozterki, czy należy je podawać na wejście sieci, bo naj-

wyżej okażą się nieprzydatne, a wtedy sieć zbuduje swoją regułę decyzyjną bez korzystania z tych „wątpliwej jakości” danych – czy też może lepiej nie „mieszać” do rozwiązania zadania zmiennych, które być może są bezwartościowe. Jaka jest Twoja opinia w tej sprawie? Odpowiedź uzasadnij.

**12. Zadanie dla zaawansowanych:** W literaturze dotyczącej sieci neuronowych czasami proponuje się, żeby struktura sieci (a więc zwłaszcza liczba warstw ukrytych i konfiguracja neuronów ukrytych) była ustalana z wykorzystaniem tak zwanego algorytmu genetycznego, naśladującego w komputerze proces biologicznej ewolucji, w wyniku której „przeżywają najlepiej przystosowani” (w tym przypadku „przeżyje” sieć mająca najlepsze właściwości w kontekście rozważanego zadania). Spróbuj dowiedzieć się więcej o tej metodzie, a potem ustal swój stosunek do niej: czy warto starać się „wyhodować drogą ewolucji” sieć o najkorzystniejszych właściwościach, czy lepiej do tej metody nie sięgać? Przedstaw argumenty przemawiające zarówno za jednym, jak i za drugim rozwiązaniem.