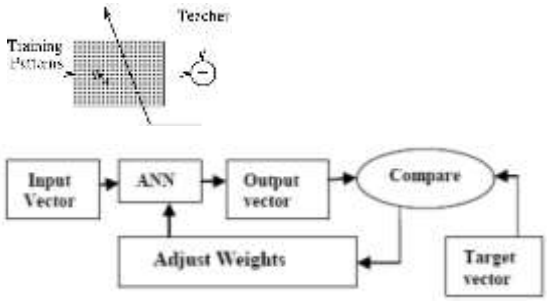
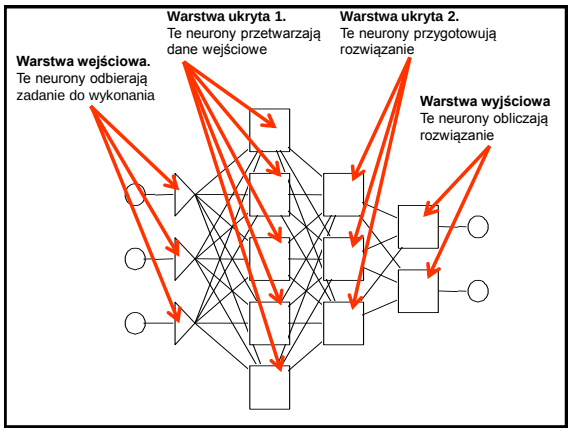
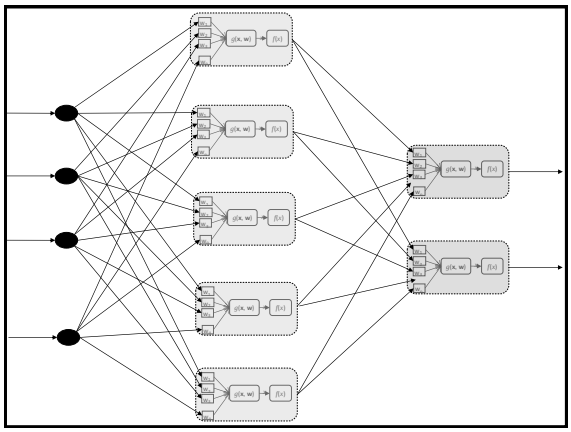
Najważniejszy element wiedzy o sieciach neuronowych

uczenie sieci

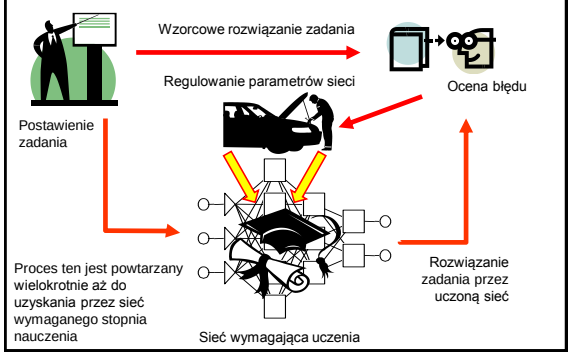
Ogólna idea procesu uczenia

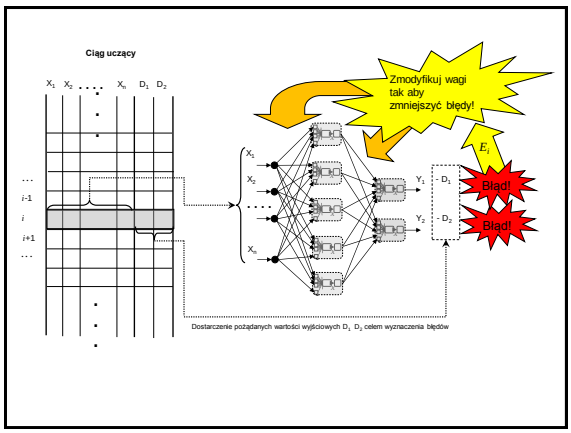






Uczenie z nauczycielem





Przykładowy zbiór uczący pozwalający rozwiązać problem regresyjny:

Jak duże będzie zanieczyszczenie powietrza w różnych miastach?

- **zmienne objaśniające:** *temperatura, liczba zakładów przemysłowych, liczba ludności, prędkość wiatru, wielkość opadu, liczba dni deszczowych*
- **zmienna objaśniana:** *Stężenie SO_2 w powietrzu*
- **dana nieistotna:** *nazwa miasta*

Uczenie sieci polega na takim doborze jej parametrów, żeby zminimalizować błąd, jaki popełnia sieć;

można to zrobić przetwarzając dane ze zbioru uczącego

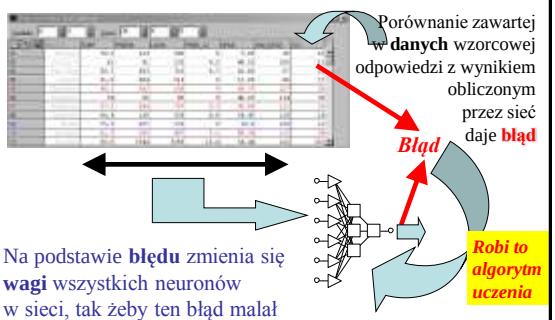
W zbiorze uczącym podane są **poprawne wartości rozwiązań** dla tych danych

Sieć na tej podstawie oblicza i podaje swoje **rozwiązanie** (na ogół początkowo błędne)

Na wejście sieci podawane są **dane** wskazane w zbiorze jako **wejściowe**

Oto sieć neuronowa, którą trzeba uczyć

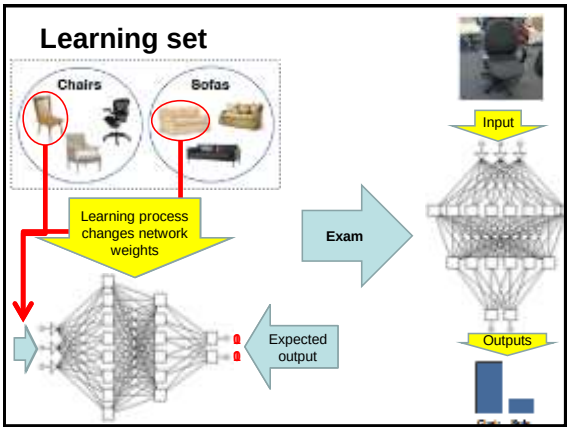
Uczenie sieci polega teraz na takim doborze jej parametrów, żeby zminimalizować błąd, jaki popełnia sieć przetwarzając dane ze zbioru uczącego



Porównanie zawartej w **danych** wzorcowej odpowiedzi z wynikiem obliczonym przez sieć daje **błąd**

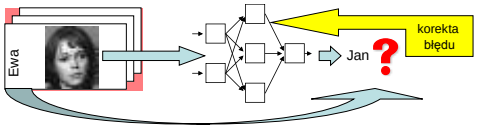
Na podstawie **błędu** zmienia się **wagi** wszystkich neuronów w sieci, tak żeby ten błąd malał

Learning set



Schemat uczenia sieci neuronowej

- uczenie jest to w istocie iteracyjny proces **estymacji** optymalnych wartości **parametrów sieci**;
- **początkowe** wartości tych parametrów są zwykle **wartościami losowymi**;
- w trakcie uczenia sieci neuronowej prezentowane są przypadki uczące i w zależności od **oceny** reakcji sieci modyfikowane są jej parametry (najczęściej **wagi synaptyczne**).



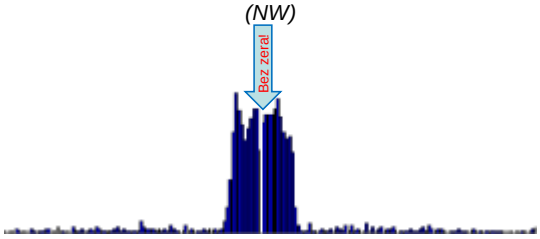
Inicjalizacja wartości wag może następować przy użyciu rozkładu Bottou

$$\left[-\frac{a}{\sqrt{n_{in}}}, \frac{a}{\sqrt{n_{in}}}\right]$$

a - uzależnione jest od rodzaju funkcji aktywacji, dla sigmoidy a=2,83;

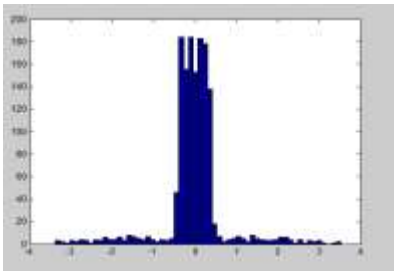
n_{in} – liczba wejść danego neuronu

Alternatywne rozwiązanie to użycie algorytmu znanego jako *Nguyen-Widrow Randomization (NW)*



Przykład rozkład uzyskany w wyniku inicjalizacji sieci algorytmem NW

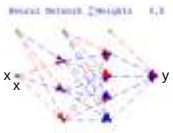
Inny przykład histogramu losowo zainicjowanych wag



Jednym ze sposobów śledzenia procesu uczenia sieci jest kontrolowanie średniej wartości modułu współczynników wagowych

Ale na ogół śledzenie
zmian wag
niewiele daje

Animowana ilustracja procesu uczenia w sieci



Istota procesu uczenia polega na modyfikacji wag połączeń neuronów

W wyniku tego sygnał wyjściowy sieci upodabnia się do zbioru uczącego

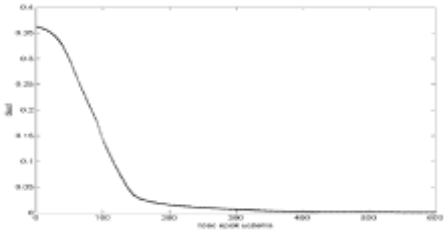
Można to obserwować biorąc pod uwagę zmiany błędu uczenia

Sieć może mieć wiele wyjść, a ponadto dla każdego zadania ze zbioru uczącego popełnia inny błąd.

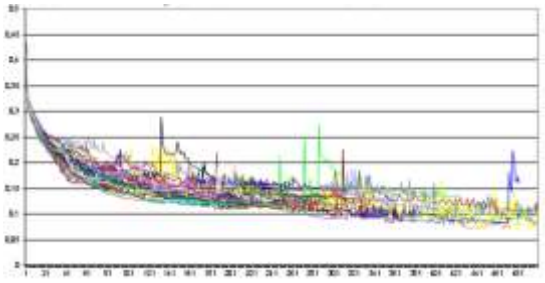
Jako syntetyczna ocena jakości działania sieci neuronowej może służyć **suma kwadratów błędów** popełnianych przez sieć na wszystkich jej wyjściach i dla wszystkich przykładów zbioru uczącego:

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

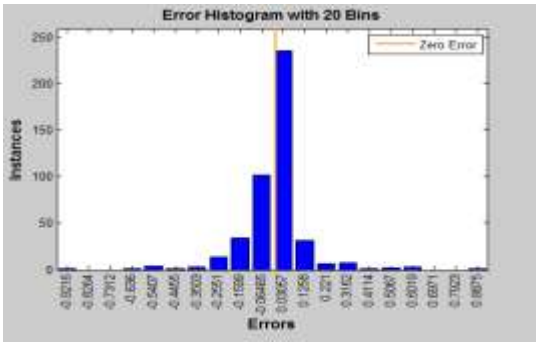
Przebieg typowej zmiany funkcji błędu w czasie procesu uczenia



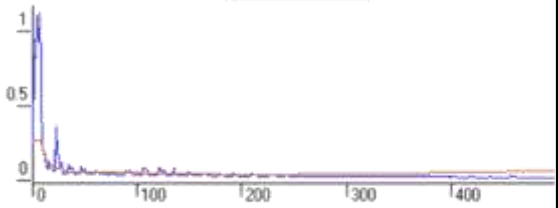
Na skutek losowego początkowego wyboru wag wielokrotne powtarzanie uczenia daje różne przebiegi krzywej zmniejszania błędu podczas uczenia, zwykle jednak są to krzywe malejące



Typowy histogram błędów sieci

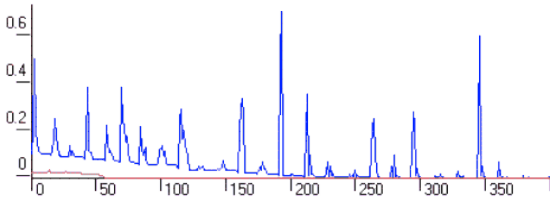


W czasie uczenia sieci obserwuje się często chwilowe pogorszenia wartości błędu

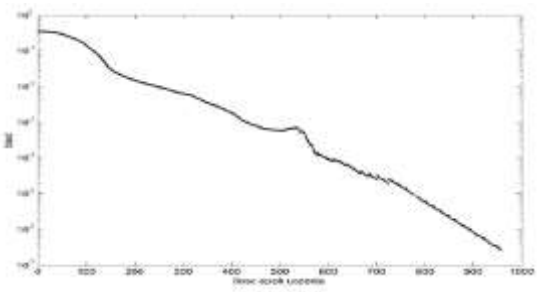


Wynika to z faktu, że w zbiorze uczącym bywają przypadki szczególnie trudne lub nietypowe, których użycie zawsze na chwilę psuje wyniki uczenia. Zjawisko to jednak zwykle szybko zanika.

Powody do zmartwienia pojawiają się dopiero wtedy, gdy takie zjawisko utrzymuje się długo i nie zanika.



Czasem wygodniej jest posługiwać się **zlogarytmowanymi** wartościami błędu uczenia, bo w takiej skali **postęp** lepiej widać zarówno na początku jak i na końcu procesu uczenia

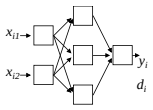


Uczenie z nauczycielem

Zbiór uczący:

X_1	X_2	D
x_{11}	x_{12}	d_1
x_{21}	x_{22}	d_2
...	...	...
x_{n1}	x_{n2}	d_n

krok uczenia numer i

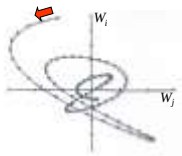


Czy $y_i = d_i$? Jeśli nie, to zmień wagi w taki sposób, aby te dwie wartości zbliżyły się do siebie.

Epoka *- Jednokrotna prezentacja wszystkich przypadków uczących*

Sposób **zmiany wag** neuronów w sieci w celu uzyskania dobrej zgodności **rzeczywistych** odpowiedzi neuronów z odpowiedziami **wzorcowymi**, podanymi w zbiorze uczącym, stanowi istotę wybranej **metody uczenia.**

Bez mieszania (randomizacji) zbioru uczącego może dochodzić do cyklicznych zmian wag podczas uczenia



Tak wygląda przebieg prawidłowego uczenia sieci (trajektoria zmiany wag)

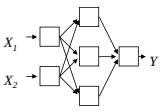
A takie przebiegi można uzyskać gdy się nie randomizuje zbioru uczącego pomiędzy kolejnymi epokami.

Cykliczność pokazywania danych uczących powoduje cykliczność zmian wartości wag, co nie posuwa naprzód procesu uczenia.

Kształtowanie się błędu sieci w trakcie uczenia

Zbiór uczący:

X_1	X_2	D
x_{11}	x_{12}	d_1
x_{21}	x_{22}	d_2
...	...	...
x_{n1}	x_{n2}	d_n



Najczęściej stosowana formuła opisująca błąd popełniany przez sieć:

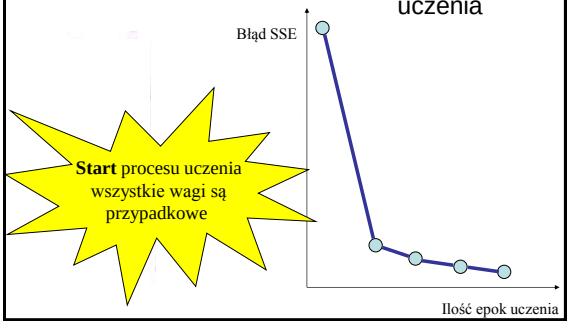
$$SSE = \sum_{i=1}^{N} (d_i - y_i)^2$$

I rzeczywiście zwykle błąd maleje!

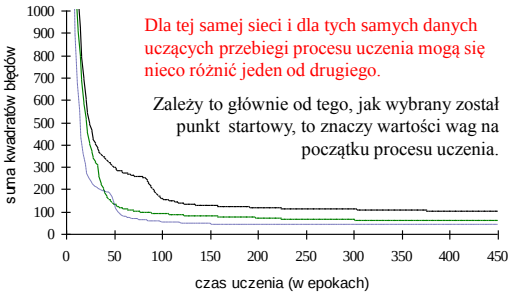
Najpierw szybko

Potem wolniej

Animowany przykładowy przebieg procesu uczenia



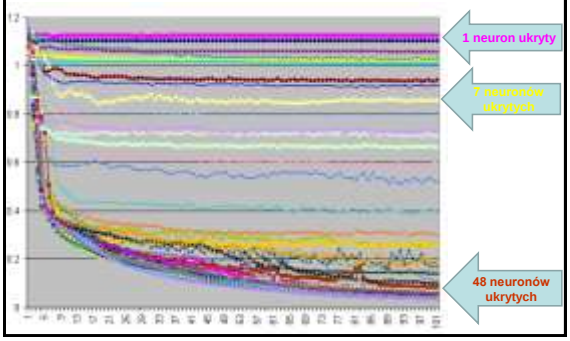
W trakcie uczenia sieci błąd przez nią popełniany zmienia się w charakterystyczny i zwykle podobny sposób: szybko maleje na początku procesu i znacznie wolniej później.



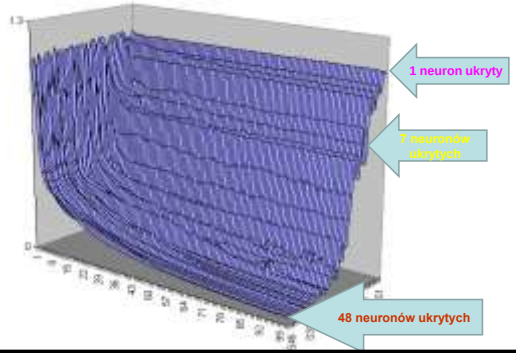
Dla tej samej sieci i dla tych samych danych uczących przebiegi procesu uczenia mogą się nieco różnić jeden od drugiego.

Zależy to głównie od tego, jak wybrany został punkt startowy, to znaczy wartości wag na początku procesu uczenia.

Uczenie sieci o różnej liczbie neuronów w warstwie ukrytej

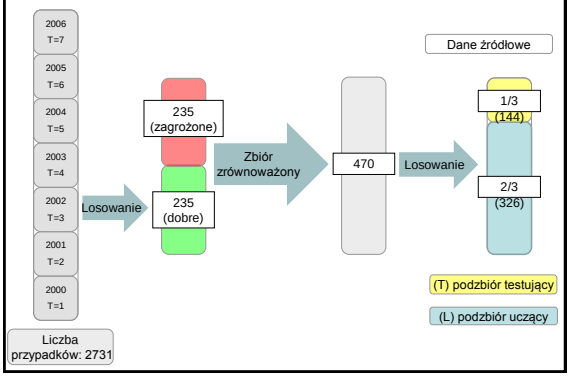


Inny sposób prezentacji uczenia sieci o różnej liczbie neuronów w warstwie ukrytej

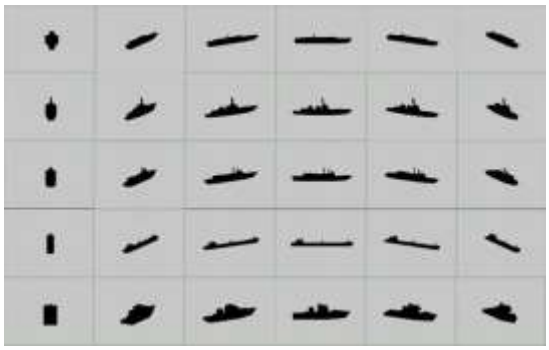


Teraz trochę szczegółów na temat zbioru uczącego

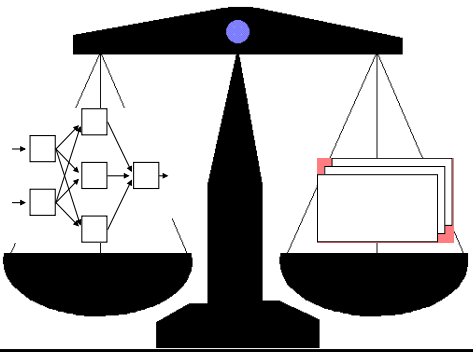
Przykład tworzenia zbioru uczącego



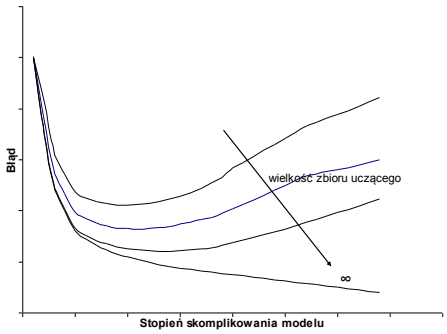
Przykładowy zbiór uczący



Trzeba także uważać na jedno: ilość informacji, jakie zdobywa (i utrwala) sieć w czasie uczenia jest **nie większa**, niż ilość informacji zawarta w zbiorze uczącym



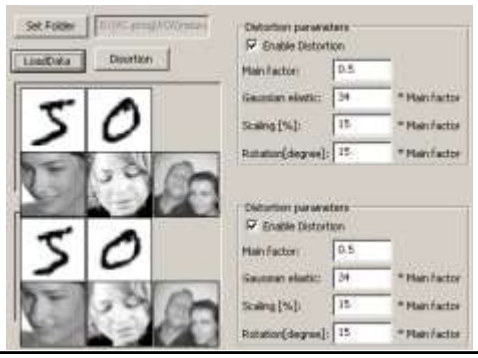
Większy zbiór uczący pozwala bezpieczniej stosować bardziej skomplikowane modele



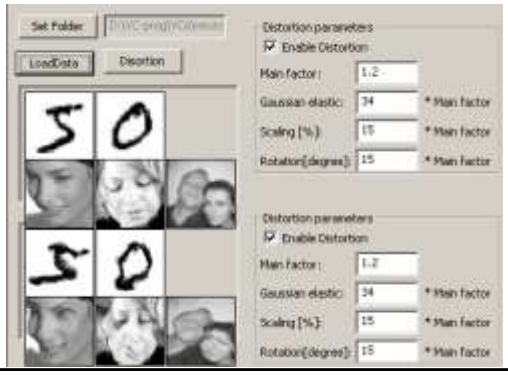
Dlatego czasem zbiór uczący się sztucznie „rozmnaża”

Inny przykład zbioru uczącego

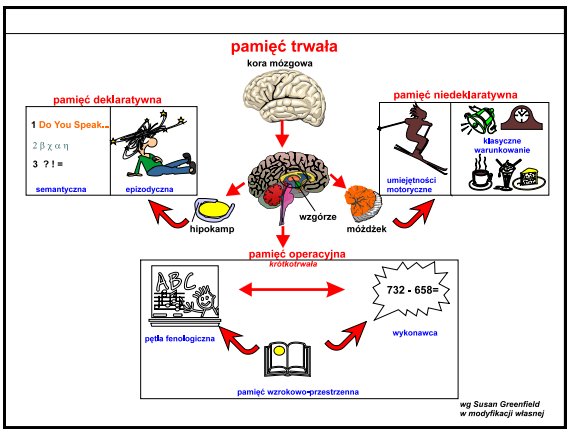
Sposób sztucznego wzbogacania zbioru uczącego poprzez drobne zniekształcenia posiadanych danych



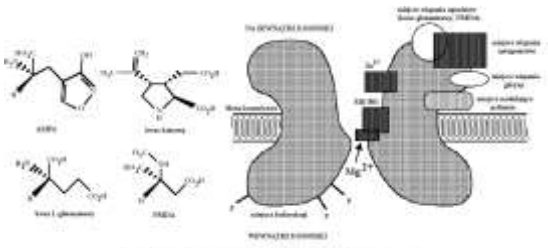
Sposób ten stosować trzeba z umiarem!



Kilka uwag o naturze biologicznej pamięci oraz procesu uczenia

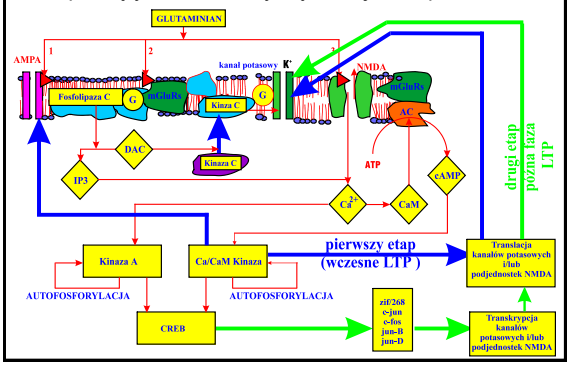


Na procesy pamięciowe duży wpływ mają zapewne receptory NMDA, spotykane w strukturach ***hipokampa*** aktywowane przez ***kwas N-metylo-D-asparaginowy***

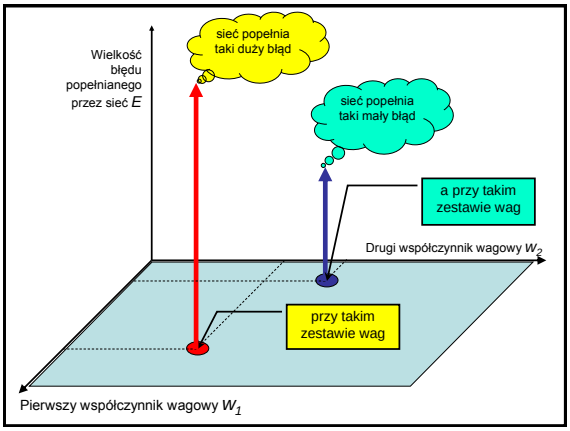


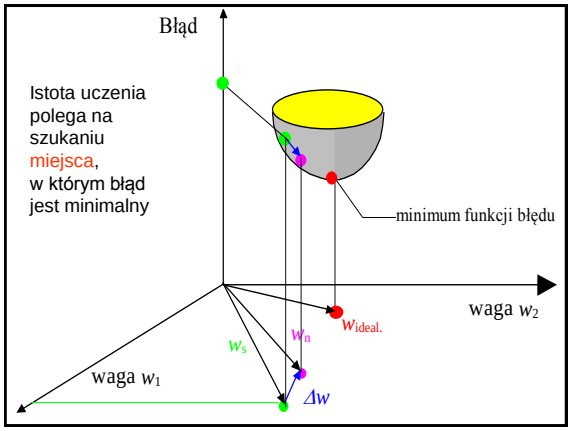
Agoniści i schemat budowy receptora NMDA

Mechanizmy biochemiczne zaangażowane w proces zapamiętywania wiedzy są raczej skomplikowane

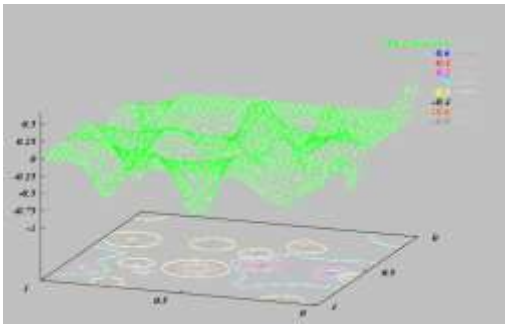


Dla uczenia sztucznych sieci neuronowych cała ta skomplikowana wiedza biochemiczna **nie jest konieczna,** bo praktyczne metody uczenia sieci oparte są na prostej intuicji:
poszukiwania minimum funkcji błędu

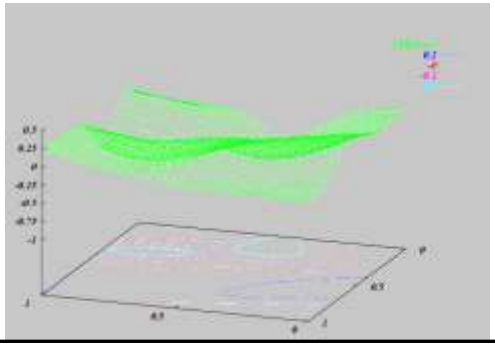




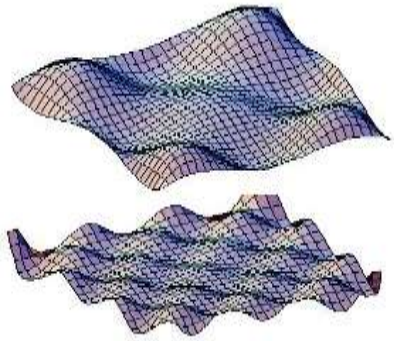
Powierzchnie błędów bywają dosyć skomplikowane!



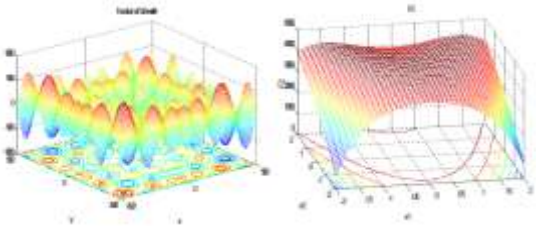
Co więcej, powierzchnia taka zmienia się radykalnie wraz ze zmianą struktury sieci!



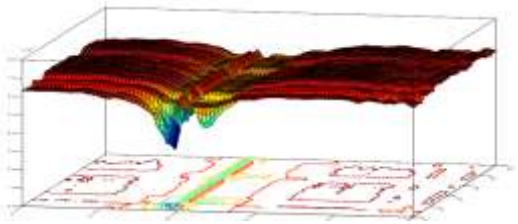
W praktycznych zadaniach powierzchnia błędu bywa mniej lub bardziej **skomplikowana**!

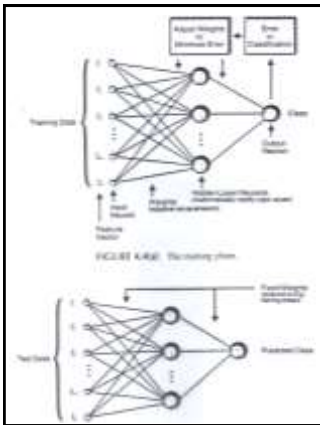


Przykłady funkcji sprawiających kłopot przy uczeniu



Inny przykład kłopotliwej funkcji

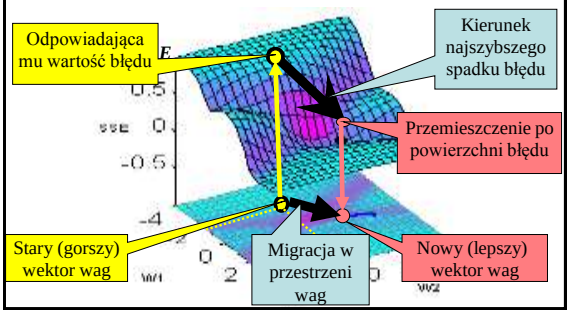




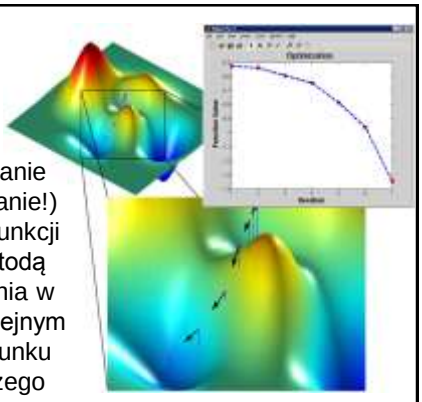
Istota procesu uczenia polega więc na **minimalizacji** funkcji:

$$E = SSE = \sum_{i=1}^{N} (d_i - y_i)^2$$

Algorytm wstecznej propagacji błędów polega na szukaniu kierunku spadku ***E*** i na takim zmienianiu wartości wag w_1, w_2, żeby zmniejszać wartość funkcji błędu w kierunku jej najszybszego spadku



Poszukiwanie (i znajdowanie!) minimum funkcji błędu metodą wyznaczania w każdym kolejnym kroku kierunku najszybszego spadku



Dane wejściowe ze zbioru uczącego

Istota algorytmu uczenia ze wsteczną propagacją błędu

Dane wejściowe ze zbioru uczącego

Istota algorytmu uczenia ze wsteczną propagacją błędu

Wynik ze zbioru uczącego

Istota algorytmu uczenia ze wsteczną propagacją błędu

Wynik ze zbioru uczącego

Istota algorytmu uczenia ze wsteczną propagacją błędu

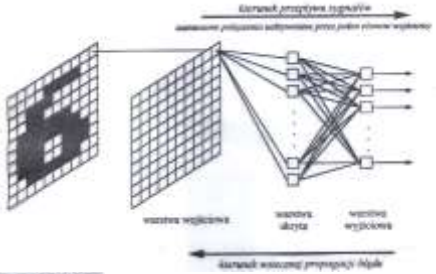
Istota algorytmu uczenia ze wsteczną propagacją błędu

Istota algorytmu uczenia ze wsteczną propagacją błędu

Istota algorytmu uczenia ze wsteczną propagacją błędu

Sieć uzyskuje nowe wagi i ma mniejszy błąd

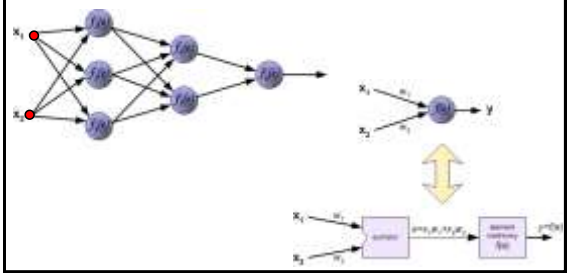
Przy uczeniu sieci najczęściej błąd wyznaczany na wyjściu sieci musi być „rzutowany" do wcześniejszych warstw, bo tam też trzeba zmieniać parametry neuronów



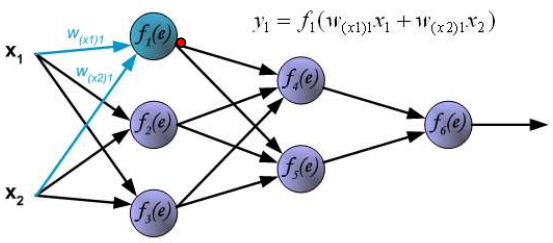
Trzeba to robić dla wszystkich elementów uczących

Uczenie sieci neuronowej krok po kroku metodą *backpropagation*

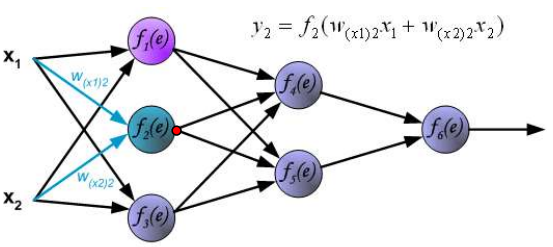
Krok 1: Podanie danych wejściowych



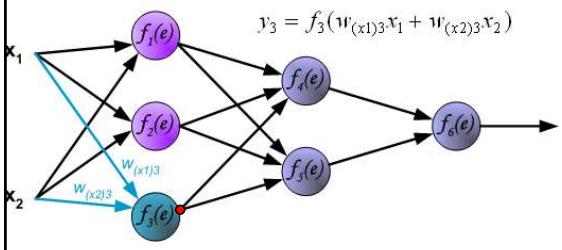
Krok 2: Obliczanie wartości wyjściowej 1. neuronu 1. warstwy ukrytej



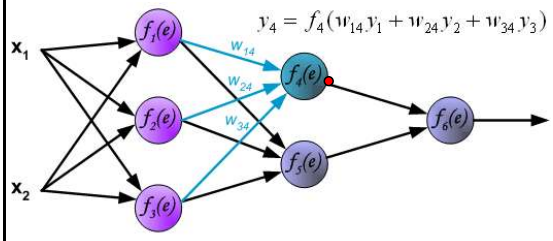
Krok 3: Obliczanie wartości wyjściowej 2. neuronu 1. warstwy ukrytej



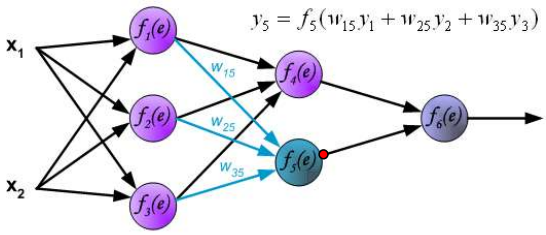
Krok 4: Obliczanie wartości wyjściowej
3. neuronu 1. warstwy ukrytej



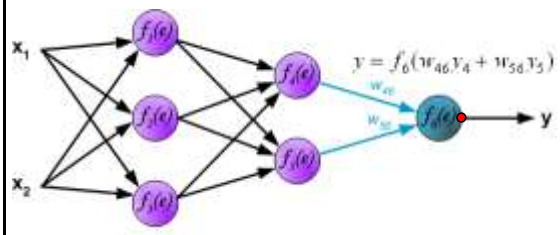
Krok 5: Obliczanie wartości wyjściowej
4. neuronu należącego do 2. warstwy ukrytej



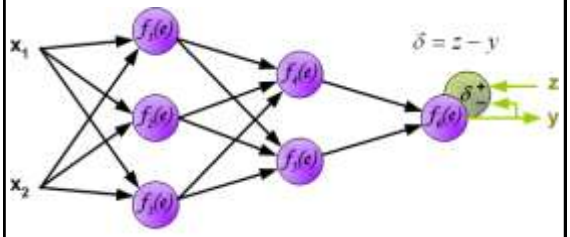
Krok 6: Obliczanie wartości wyjściowej
5. neuronu należącego do 2. warstwy ukrytej



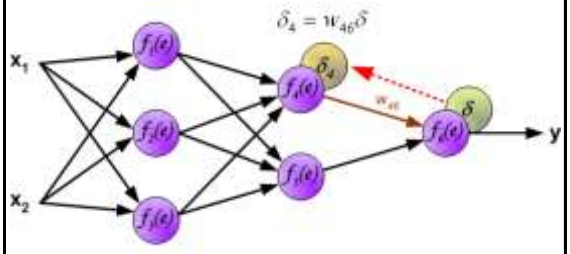
Krok 7: Obliczanie wartości wyjściowej
z całej sieci przez neuron wyjściowy



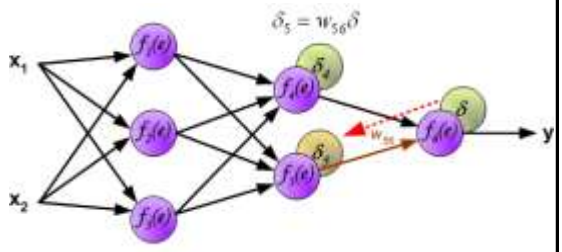
Krok 8: Obliczanie błędu wartości wyjściowej
z całej sieci



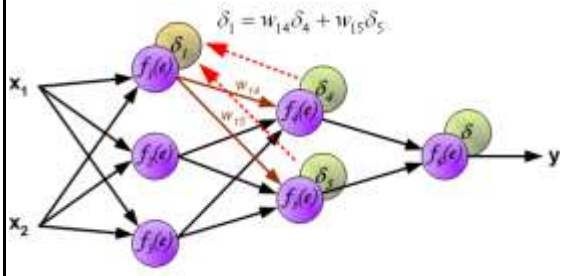
Krok 9: Obliczanie błędu 4. neuronu
należącego do 2. warstwy ukrytej



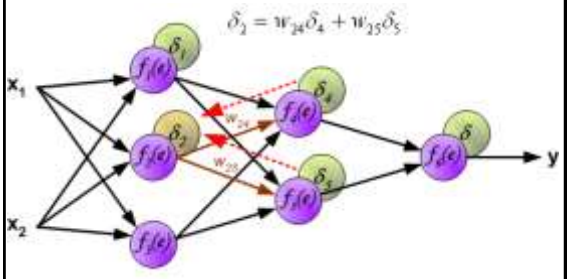
Krok 10: Obliczanie błędu 5. neuronu należącego do 2. warstwy ukrytej



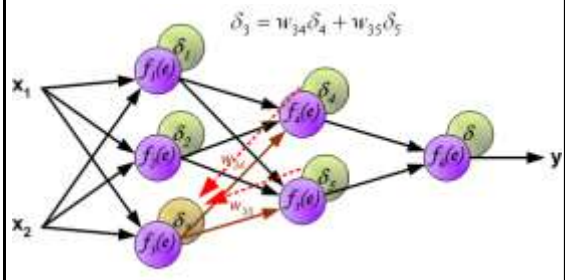
Krok 11: Obliczanie błędu 1. neuronu należącego do 1. warstwy ukrytej



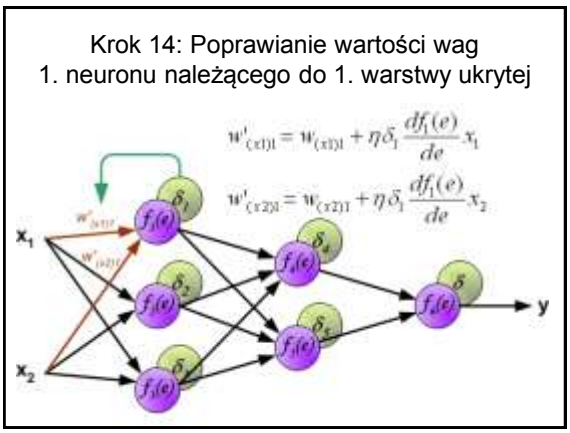
Krok 12: Obliczanie błędu 2. neuronu należącego do 1. warstwy ukrytej



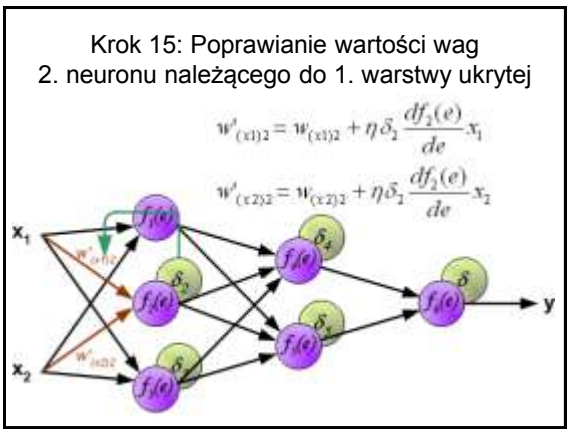
Krok 13: Obliczanie błędu 3. neuronu należącego do 1. warstwy ukrytej



Krok 14: Poprawianie wartości wag 1. neuronu należącego do 1. warstwy ukrytej



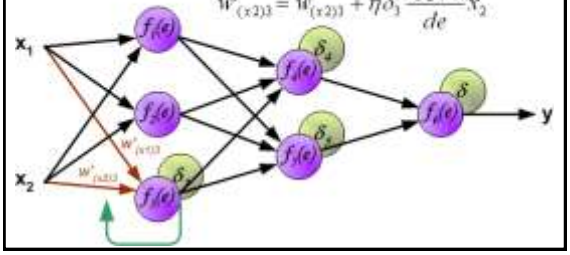
Krok 15: Poprawianie wartości wag 2. neuronu należącego do 1. warstwy ukrytej



Krok 16: Poprawianie wartości wag
3. neuronu należącego do 1. warstwy ukrytej

$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$$

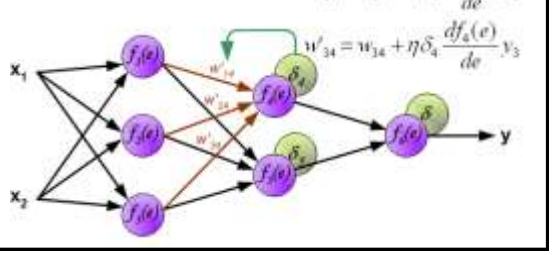


Krok 17: Poprawianie wartości wag
4. neuronu należącego do 2. warstwy ukrytej

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$

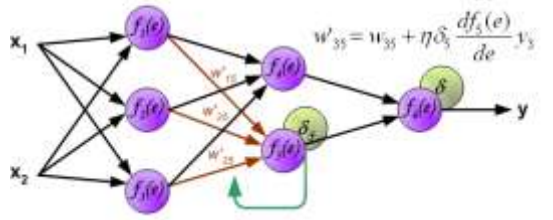


Krok 18: Poprawianie wartości wag
5. neuronu należącego do 2. warstwy ukrytej

$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2$$

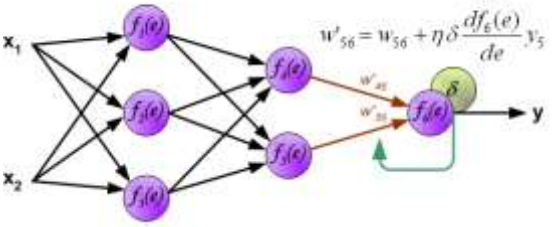
$$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$



Krok 19: Poprawianie wartości wag
6. neuronu należącego do warstwy wyjściowej

$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5$$



Krok 20: A teraz te wszystkie czynności trzeba powtórzyć jeszcze raz (od początku do końca!) dla następnego elementu zbioru uczącego - i tak dalej …

We wszystkich przytaczanych wyżej wzorach występował współczynnik uczenia η.

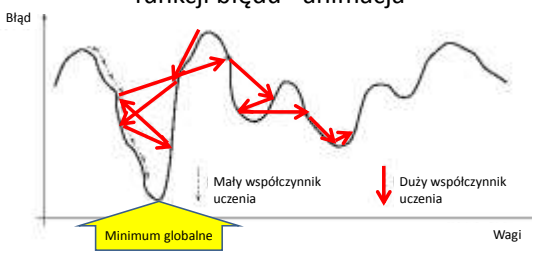
Jego właściwy dobór to też jeden z problemów koniecznych do rozwiązania

Im współczynnik η jest większy, tym większa poprawka wag jest wykonywana przy takiej samej wartości błędu.

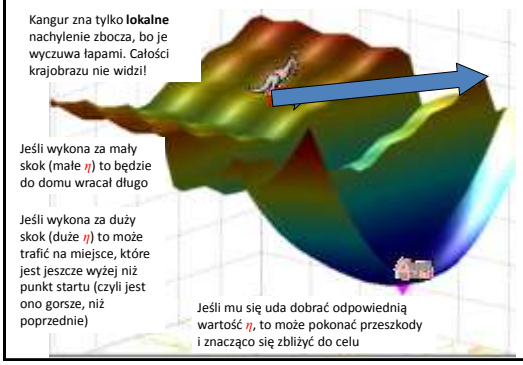
Algorytm z dużym η można kojarzyć z surowym i wymagającym nauczycielem, który po każdym błędzie ucznia zmusza go do radykalnej zmiany postępowania.

Jest to również miara przeniesienia **lokalnej** własności funkcji błędu wyznaczonej w skali mikro (gradient określa kierunek zmian wag zapewniający malenie błędu tylko w przypadku **nieskończenie małych** zmian wag) **na działanie w skali makro** (korekta wag dokonywana jest z pewnym **nie nieskończenie małym** krokiem, którego wielkość wyznacza η)

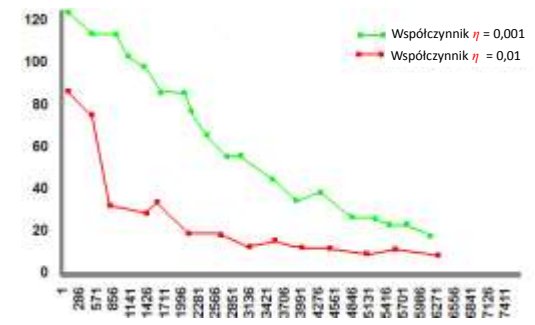
Problem doboru wartości współczynnika uczenia η w przypadku jednowymiarowej funkcji błędu - animacja



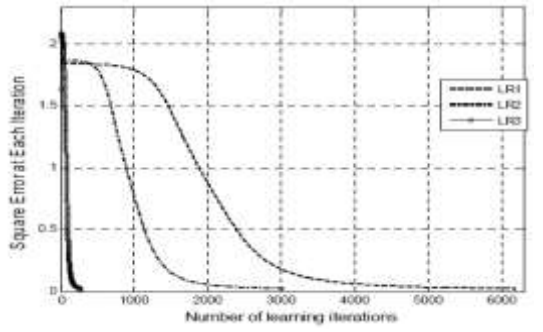
Uczenie sieci neuronowej jako wędrówka ślepego kangura szukającego w górach domu który jest w dolinie



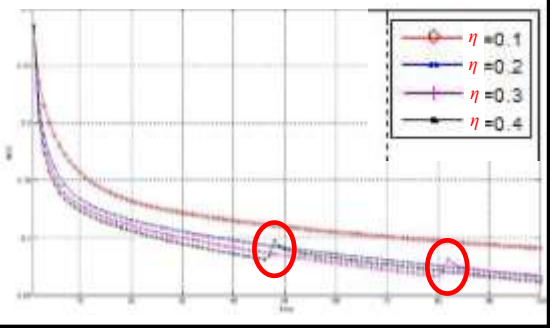
Wpływ współczynnika uczenia η na przebieg treningu sieci neuronowej. Najczęściej większa wartość przyspiesza uczenie i polepsza efekt



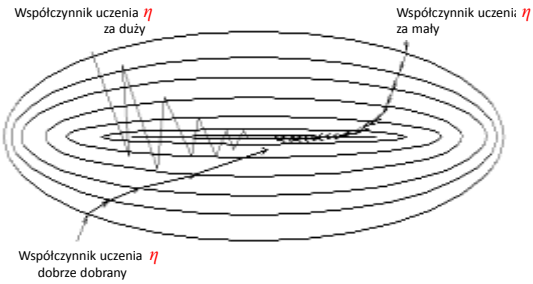
Zależność przebiegu uczenia od wartości współczynnika uczenia η oznaczanego także jako LR (*Learning Rate*)



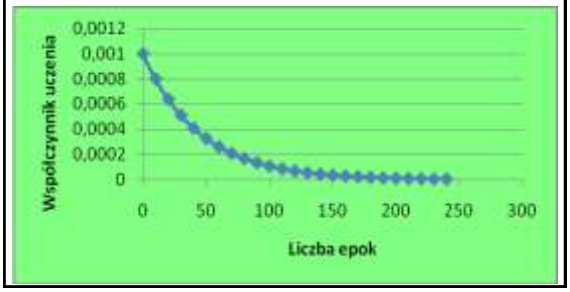
Ale przy za dużych wartościach η pojawia się niestabilność



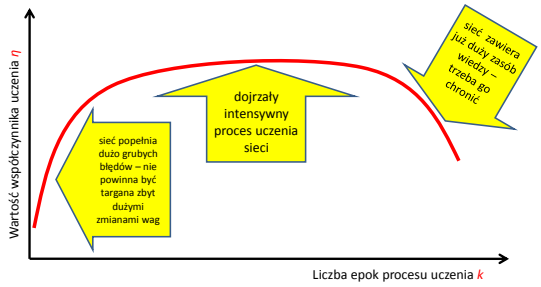
Wpływ doboru η na przebieg uczenia



W niektórych pracach zaleca się stopniowe zmniejszanie współczynnika uczenia w trakcie uczenia



Optymalne wydają się jednak **niemonotoniczne** zmiany wartości współczynnika uczenia w trakcie nauki



Dla ustabilizowania procesu uczenia stosuje się współczynnik wnoszący **bezwładność** – zwany *momentum*

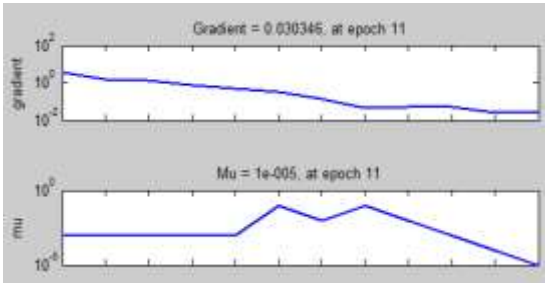
$$W_{ij}^{l}(k+1) = W_{ij}^{l}(k) + \eta\frac{\partial E(W)}{\partial W_{ij}^{l}} + \beta(W_{ij}^{l}(k) - W_{ij}^{l}(k-1))$$

Interpretacja roli *momentum*

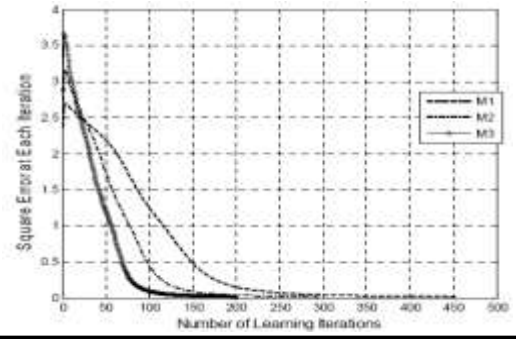
Lekkie małe saneczki utkną w pierwszym dołku na stoku

Rozpędzony masywny bojer ześlizgnie się aż na sam dół!

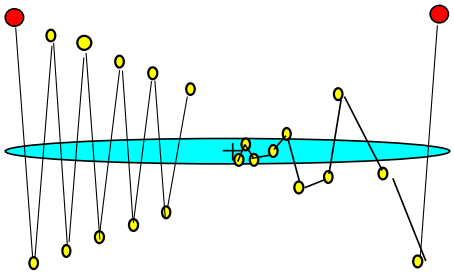
Przebieg wartości gradientu i momentum jako sił napędowych procesu uczenia w przykładowym eksperymencie uczenia sieci neuronowej



Zależność przebiegu uczenia od wartości współczynnika *momentum* M

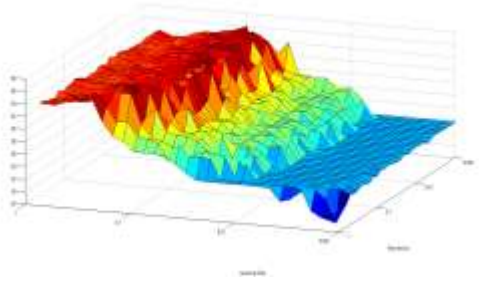


Zmiany współczynników wag w sieci podczas uczenia bez współczynnika momentum (po lewej) i ze współczynnikiem momentum równym 0.5 (po prawej) dla tej samej ilości kroków

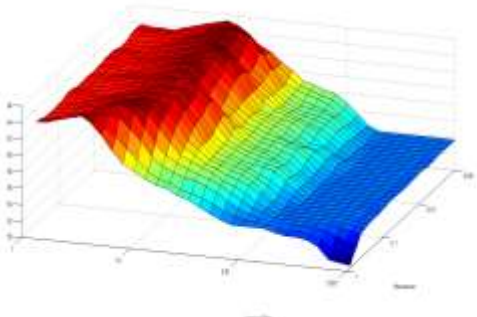


Momentum przyspiesza i stabilizuje uczenie!

Łączny dobór wartości współczynnika uczenia i momentum dane surowe

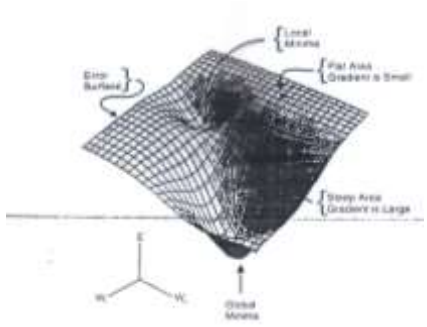


Łączny dobór współczynnika uczenia i momentum dane wygładzone

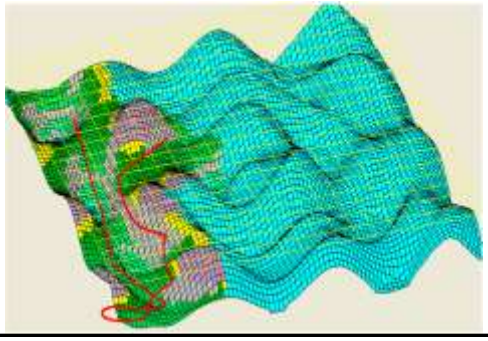


Trochę szczegółów na temat przebiegu procesu uczenia

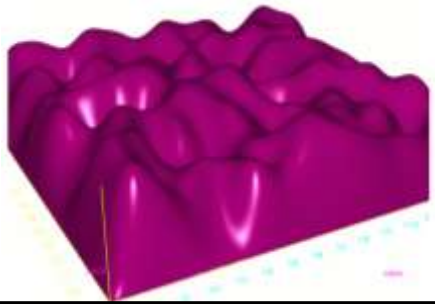
Dlatego proces uczenia może napotykać na różne przeszkody:



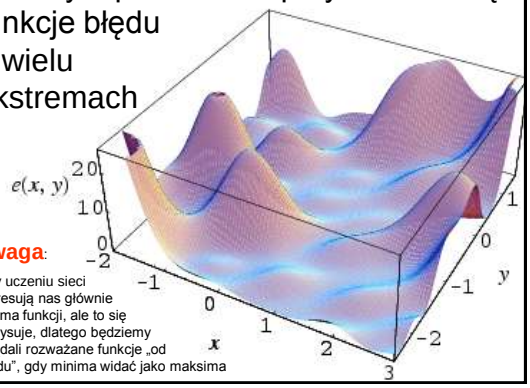
Pułapką dla procesu uczenia mogą być także długie wąskie „wąwozy”



Przykładowy obraz **rzeczywistej** powierzchni błędu powstającej przy uczeniu sieci neuronowej. W jednej z dolin kryje się globalne minimum. Tylko w której?!



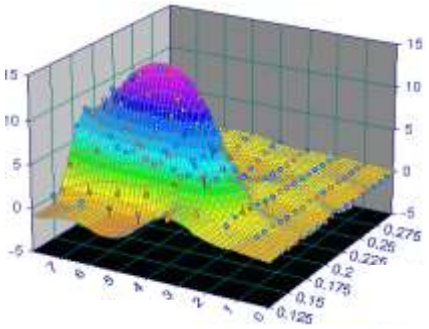
Głównym problemem przy uczeniu są funkcje błędu o wielu ekstremach



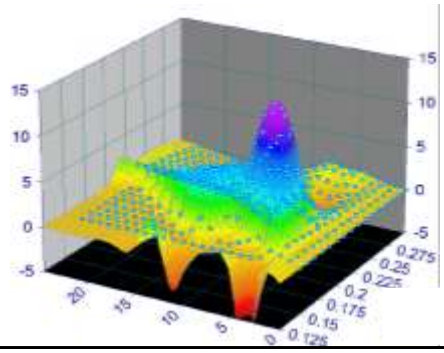
Uwaga:

Przy uczeniu sieci interesują nas głównie minima funkcji, ale to się źle rysuje, dlatego będziemy oglądali rozważane funkcje „od spodu”, gdy minima widać jako maksima

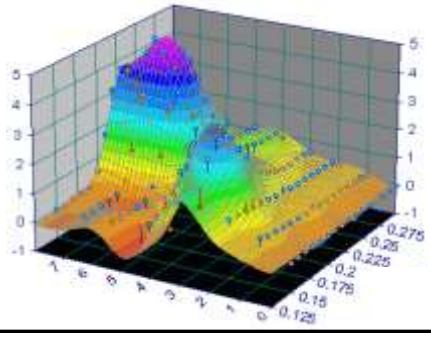
Bywają zadania, dla których jest tylko jedno ekstremum funkcji błędu. Jego znalezienie jest wtedy łatwe



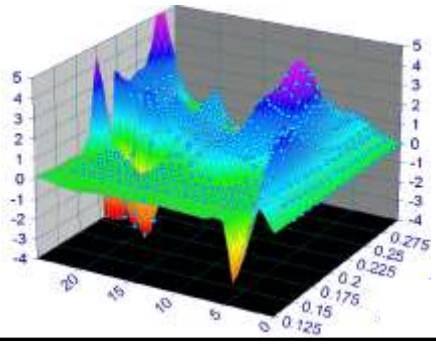
Czasem funkcja błędu ma więcej szczegółów topologicznych, ale wśród nich tylko jedno dobrze zarysowane ekstremum



Nie jest najgorzej, jeśli z ekstremum globalnym konkuruje tylko jedno ekstremum lokalne o wyraźnie odróżnialnych własnościach

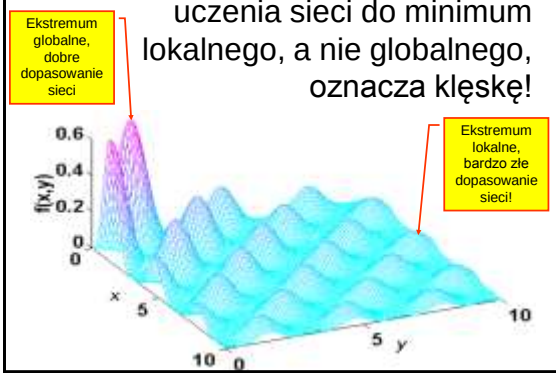


Często jednak lokalnych minimów jest wiele, a każde z nich może być pułapką dla procesu uczenia

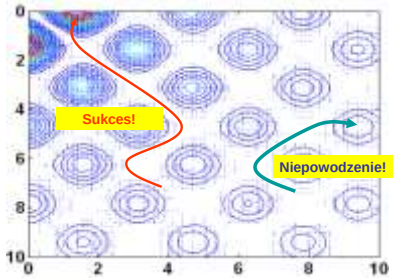


Problem minimów lokalnych

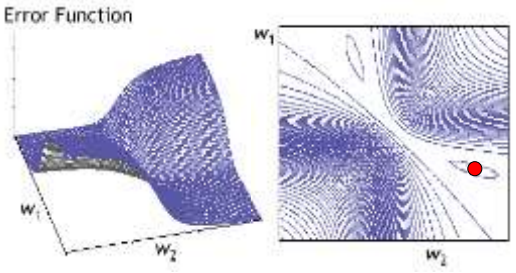
Dla takiej funkcji trafienie podczas uczenia sieci do minimum lokalnego, a nie globalnego, oznacza klęskę!



Ta sama funkcja w postaci mapy pozwala ocenić trajektorie procesu uczenia, która powinna trafić do minimum globalnego



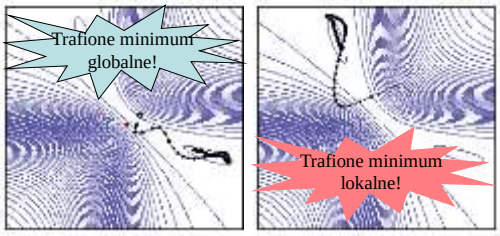
W dalszych przykładach będziemy używali funkcji błędu mającej jedno minimum lokalne i jedno globalne



Oto wykres tej funkcji...

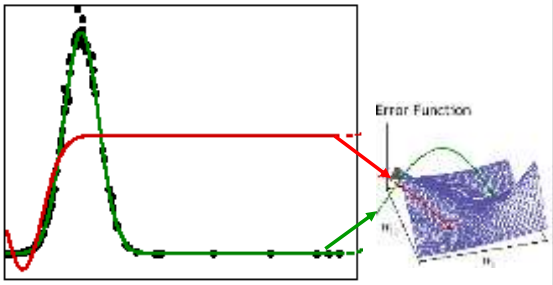
oraz jej mapa z „poziomicami”

Wynik uczenia zależy od wyboru początkowych wartości wag (punktu startowego algorytmu)

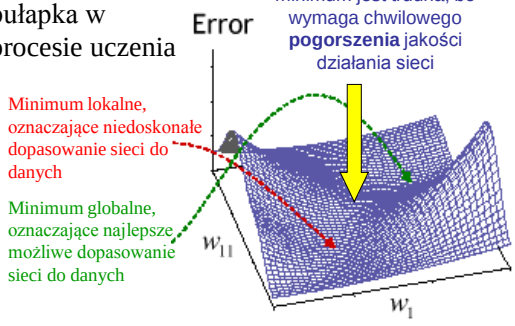


Jest to ważne dla praktyki, ponieważ podczas uczenia zawsze zagrażają minima lokalne funkcji błędu, w których rozwiązanie problemu jest bardzo **niedoskonałe**!

Utknięcie procesu uczenia w „pułapce” minimum lokalnego wyjaśnia, dlaczego czasem nauczona sieć dopasowuje swoje działanie tylko do części danych



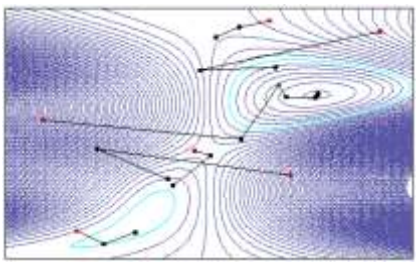
Minimum lokalne jako pułapka w procesie uczenia



Minima lokalne funkcji błędu utrudniają prawidłowe przeprowadzenie uczenia

- Przeciwdziałanie polega na wprowadzeniu dodatkowego czynnika losowego do uczenia:
  - losowe zaburzenia wartości wejściowych (*Noise*),
  - zmiana kolejności prezentacji przykładów (*Shuffle Cases*),
  - losowe zaburzenie wartości wag (*Jog Weights*),
  - wielokrotne powtarzanie procesu uczenia (od losowo wybranych punktów startowych)

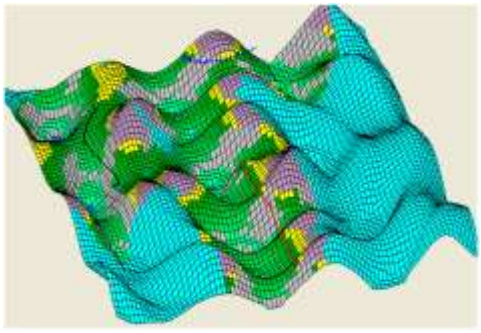
Zwykle kilka prób uczenia sieci startujących **z różnych punktów początkowych** pozwala wykryć, gdzie jest poszukiwane minimum globalne



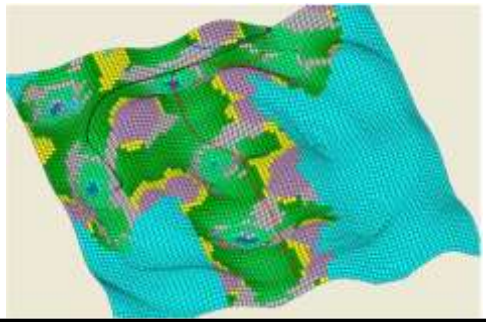
Zestawy wag uzyskiwane przy powtarzaniu uczenia tej samej sieci na tych samych danych często się powtarzają, ale to wcale nie musi być regułą

Pierwszy proces uczenia	Drugi proces uczenia
Trzeci proces uczenia	Czwarty proces uczenia

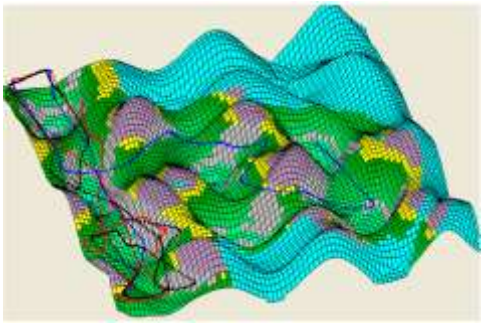
Dlatego startując z różnych punktów docieramy do różnych minimów



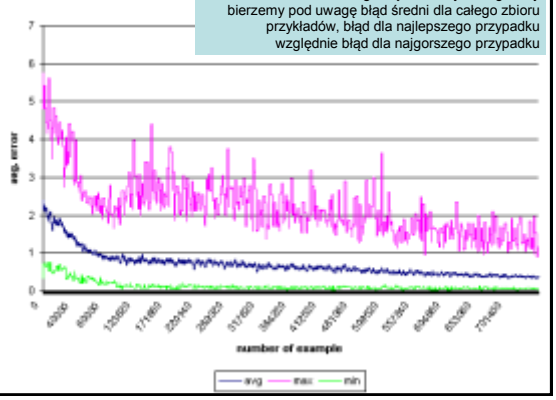
Nawet przy starcie z identycznego punktu początkowego przebieg powtórnego uczenia sieci neuronowej może prowadzić do innych wyników, bo kolejność prezentacji elementów zbioru uczącego jest losowa



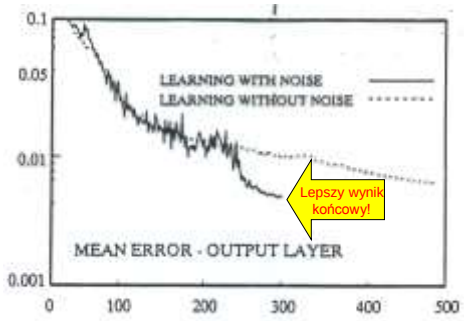
Wielokrotne ponawianie procesu uczenia może pomóc w pokonaniu także pułapki wąskich wąwozów

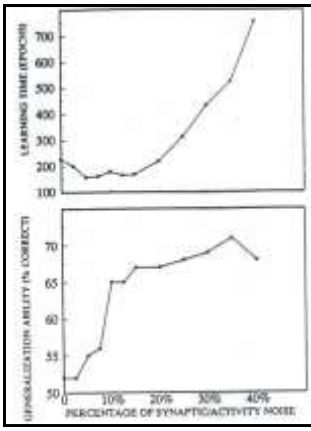


Wielkość obserwowanego błędu zależy od tego, czy bierzemy pod uwagę błąd średni dla całego zbioru przykładów, błąd dla najlepszego przypadku względnie błąd dla najgorszego przypadku



Wpływ zakłóceń w danych uczących na proces uczenia



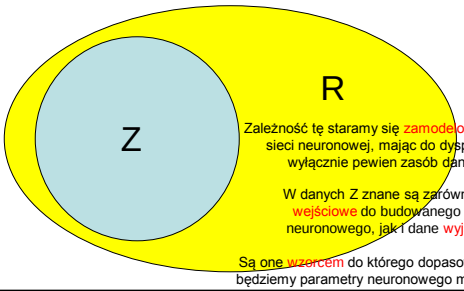


Dodanie szumu do danych uczących wydłuża wprawdzie czas uczenia, ale powiększa zdolność sieci do generalizacji wyników uczenia

Na jakiej podstawie przerwać uczenie sieci?

Podział zbiorów danych przy uczeniu sieci

Celem użycia sieci jest odtworzenie pewnej nieznanej zależności, która ma funkcjonować dla wszystkich danych należących do pewnego zbioru R.

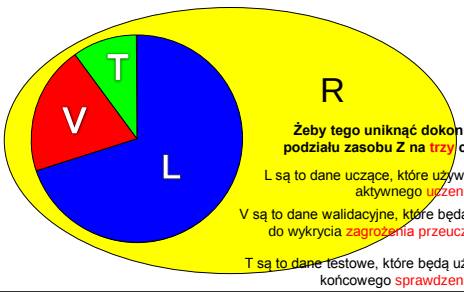


Zależność tę staramy się zamodelować w sieci neuronowej, mając do dyspozycji wyłącznie pewien zasób danych Z.

W danych Z znane są zarówno dane wejściowe do budowanego modelu neuronowego, jak i dane wyjściowe.

Są one wzorcem do którego dopasowywać będziemy parametry neuronowego modelu.

Niebezpieczeństwo, które nam zagraża, polega na tym, że podczas uczenia sieci może dojść do nadmiernego dopasowania modelu do właściwości danych ze zbioru Z, nie koniecznie reprezentujących wszystkie dane w zbiorze R.



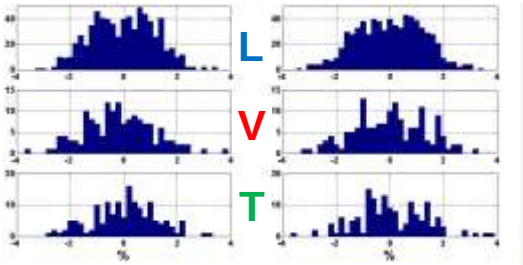
Żeby tego uniknąć dokonujemy podziału zasobu Z na trzy części:

L są to dane uczące, które używamy do aktywnego uczenia sieci.

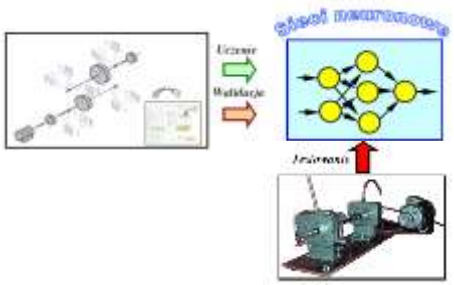
V są to dane walidacyjne, które będą użyte do wykrycia zagrożenia przeuczeniem

T są to dane testowe, które będą użyte do końcowego sprawdzenia sieci

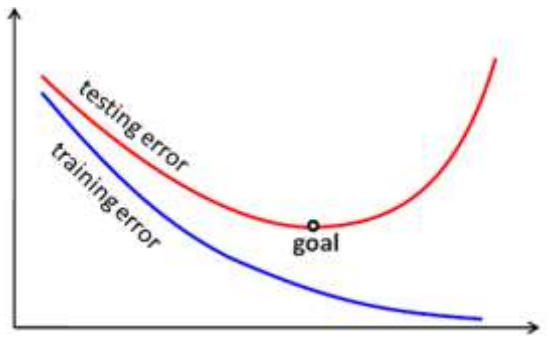
Przykładowe rozkłady błędów dla dwóch różnych sieci neuronowych odpowiednio dla zbiorów **L**, **V** i **T**.



Jeśli to tylko możliwe, to należy dążyć do tego, żeby testowanie przeprowadzać na innych danych, niż były używane do uczenia i walidacji



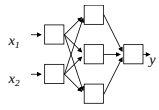
Przeuczenie jako problem



Przeuczenie sieci czyli zanik zdolności do generalizacji

Zbiór uczący:

X_1	X_2	D
x_{11}	x_{12}	d_1
x_{21}	x_{22}	d_2
...	...	...
x_{k1}	x_{k2}	d_k



W trakcie uczenia obserwujemy:

Zdolność do aproksymacji:

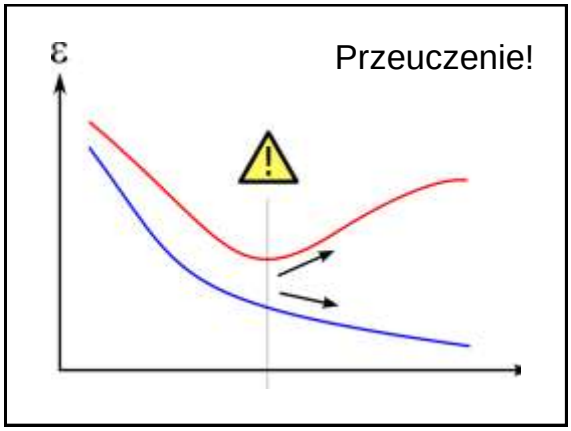
$$SSE_U = \sum_{i=1}^{k} (d_i - y_i)^2$$

Zdolność do generalizacji:

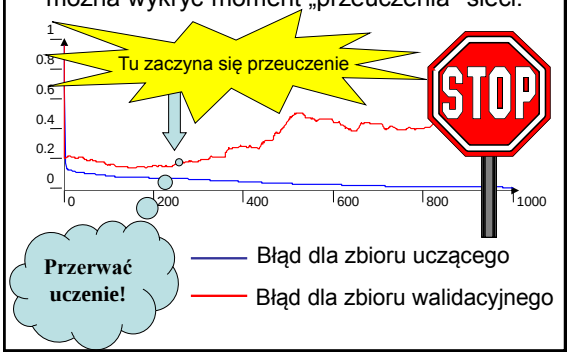
$$SSE_W = \sum_{i=k+1}^{n} (d_i - y_i)^2$$

Zbiór walidacyjny:

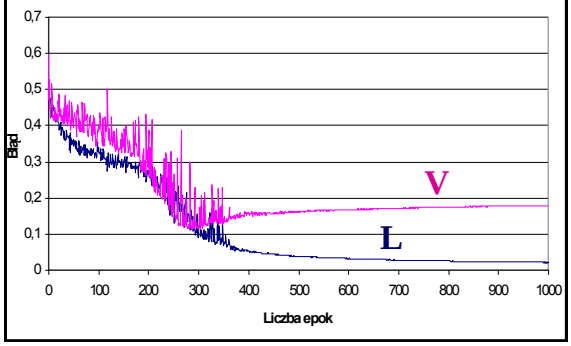
X_1	X_2	D
$x_{(k+1)1}$	$x_{(k+1)2}$	d_{k+1}
...	...	...
x_{n1}	x_{n2}	d_n



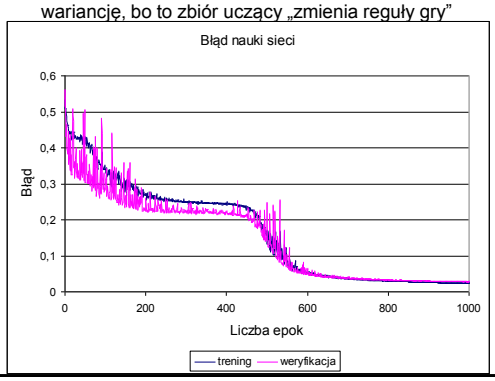
Dzięki użyciu zbioru danych walidacyjnych można wykryć moment „przeuczenia” sieci.



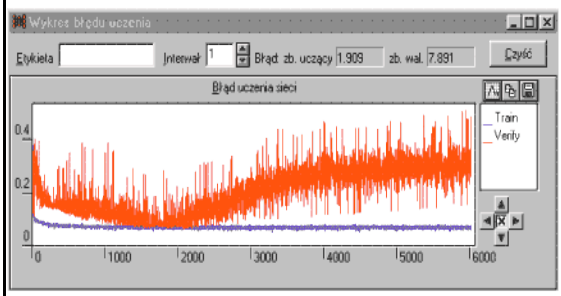
Na zbiorze **V** uzyskuje się zwykle gorsze wartości błędu, niż na zbiorze **L**



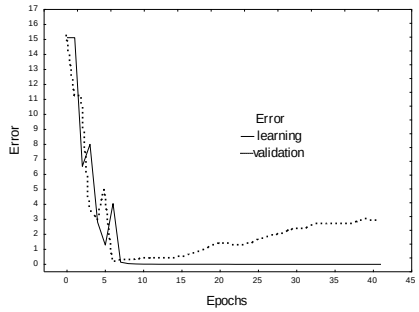
Bywa, że błąd na zbiorze walidacyjnym ma **mniejszą** wartość niż na zbiorze uczącym, ale z reguły ma dużą wariancję, bo to zbiór uczący „zmienia reguły gry”



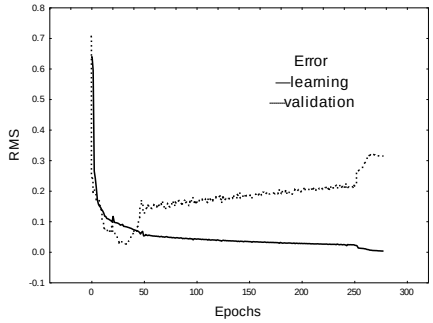
Ale czasem efekt przeuczenia jest ewidentny



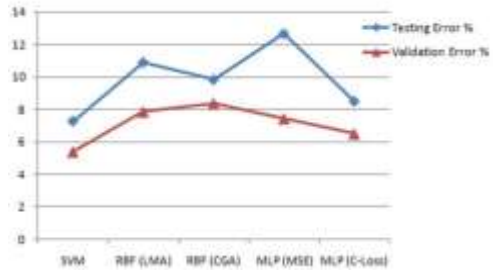
Inny rzeczywisty przykład przeuczenia



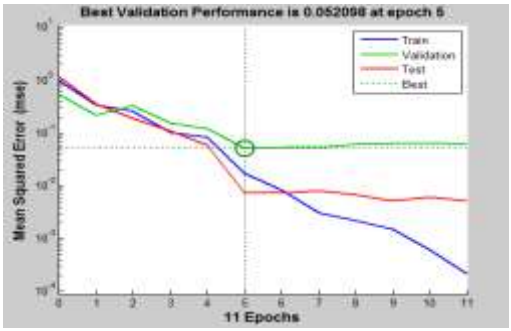
Uczenie z wyraźnym niemal skokowym pogorszeniem generalizacji w wyniku przeuczenia



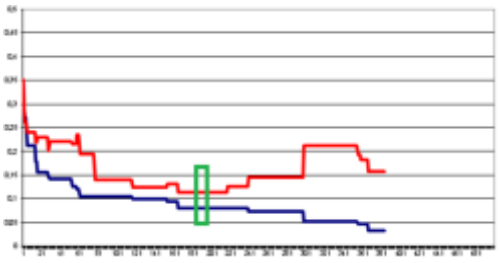
Jeśli stać nas na wydzielenie osobnych danych testowych to można sprawdzić, czy błąd uzyskany przy walidacji nie był wynikiem zbiegu okoliczności

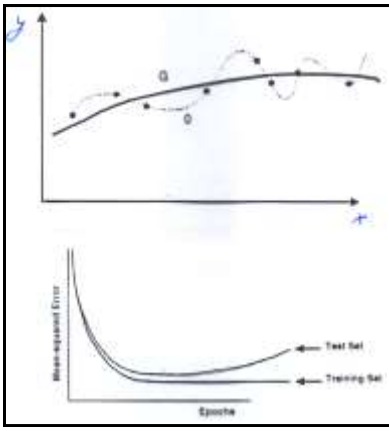


Wprowadzenie oceny na zbiorze testowym



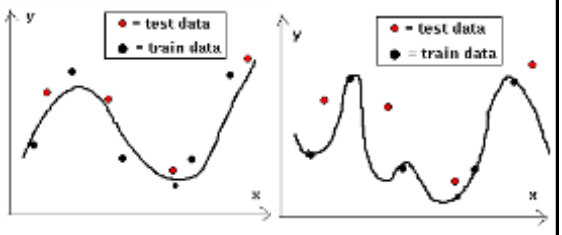
Co ciekawe – efekt przeuczenia daje się zauważyć także wtedy, gdy uczenie sieci przeprowadzane jest metodą ewolucyjną (algorytmem genetycznym)





Skąd się bierze utrata zdolności do generalizacji?

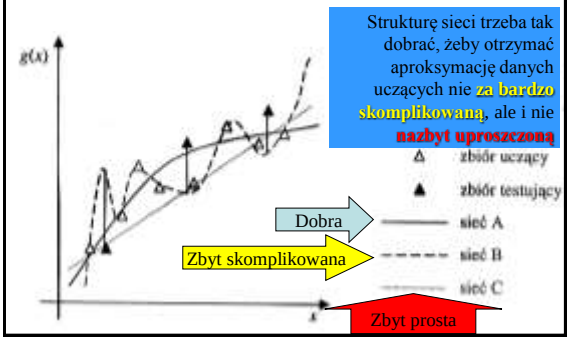
Na tych samych danych uczących sieć może zbudować model dobrze albo źle generalizujący



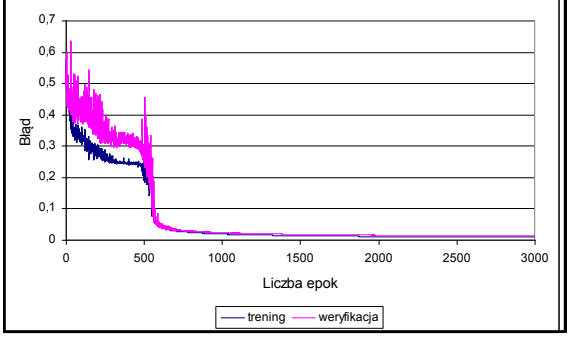
Model zachowujący dobrą zdolność generalizacji

Model który utracił zdolność generalizacji

Jedną z głównych przyczyn złej generalizacji jest niewłaściwa struktura sieci – za bardzo lub za mało skomplikowana

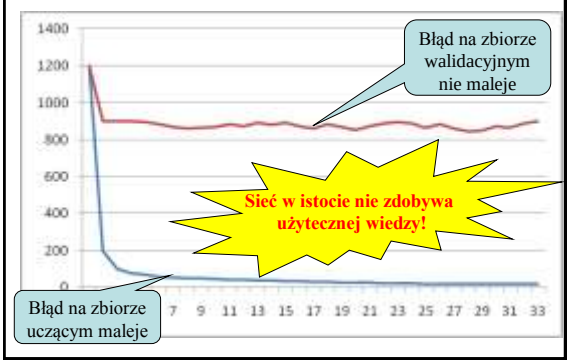


Uczenie „z trudnym rozruchem”

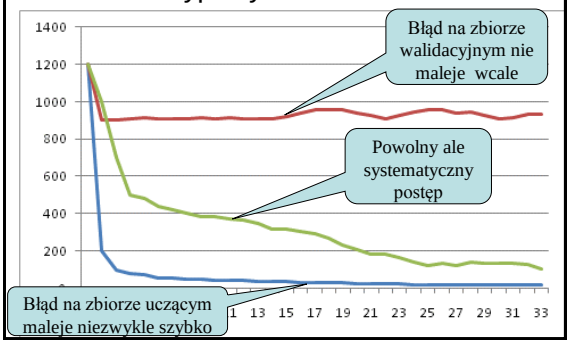


Problem z sieciami, które są nadmiernie inteligentne

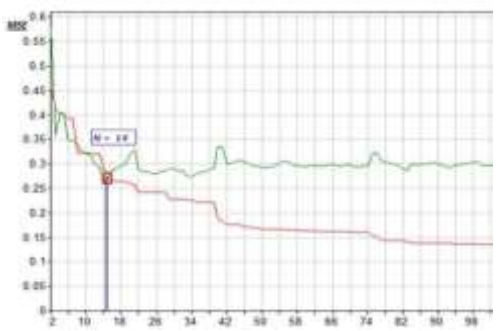
Efekt „uczenia na pamięć”



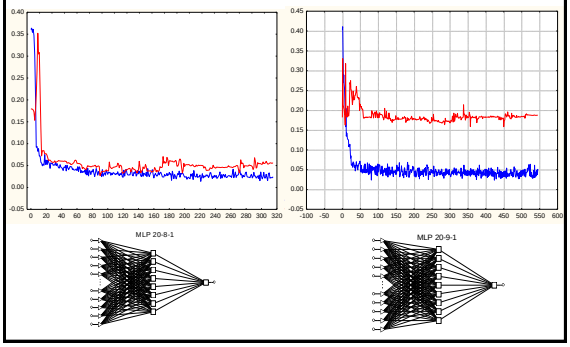
Efekt uczenia na pamięć w zestawieniu z typowym uczeniem



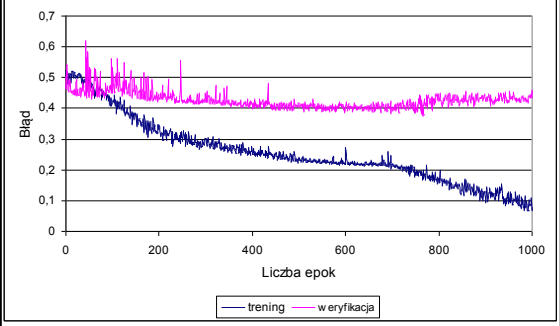
Przykład sieci, która w pewnym momencie zaczęła uczyć się na pamięć



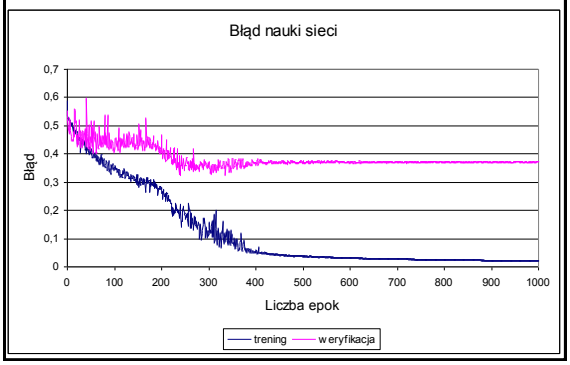
Czasem dodanie **jednego** neuronu do warstwy ukrytej sieci może „przerzucić" sieć z trybu uczenia z uogólnianiem do trybu uczenia na pamięć



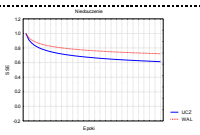
Inny efekt „uczenia na pamięć"



Jeszcze jeden przykład uczenia się na pamięć



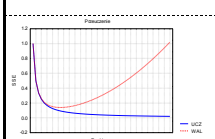
Efekty uczenia sieci - 1



Przypadek I

Błąd wyznaczony na podstawie zbioru uczącego oraz błąd wyznaczony na podstawie zbioru walidacyjnego maleją, ale pozostają pod koniec uczenia na wysokim poziomie. Przyczyną takiego zachowania może być czasem zbyt krótki czas uczenia sieci ale najczęściej przyczyną jest zbyt mała liczba elementów ukrytych sieci.

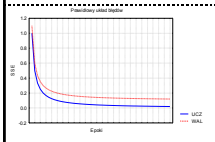
Efekty uczenia sieci - 2



Przypadek II

Błąd wyznaczony na podstawie zbioru uczącego bardzo szybko się zmniejsza, zaś błąd walidacyjny początkowo również maleje, a następnie zaczyna się zwiększać. Taki sposób kształtowania się wartości błędów jest klasycznym objawem przeuczenia. Jego podstawową przyczyną jest zbytnio rozbudowana struktura sieci.

Efekty uczenia sieci - 3

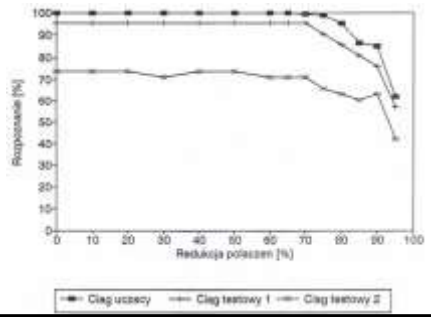


Przypadek III

Zarówno wartość błędu wyznaczona na podstawie zbioru uczącego, jak i zbioru walidacyjnego ulega wyraźnemu zmniejszaniu w trakcie uczenia i osiąga akceptowalny przez użytkownika poziom. Poziom błędu walidacyjnego jest zwykle nieznacznie wyższy od jego odpowiednika obliczonego na podstawie zbioru uczącego. Strukturę sieci i przebieg uczenia należy ocenić prawidłowo.

Dodatkowe ciekawe właściwości sieci neuronowych

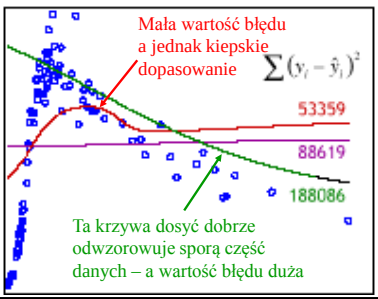
Nauczona sieć neuronowa wykazuje ograniczoną wrażliwość na uszkodzenia jej elementów („redukcja połączeń")



Ewaluacja i poprawa jakości modelu neuronowego

Przy optymalizacji parametrów sieci nie mamy lepszego kryterium, niż minimalizacja błędu ***E***

Trzeba jednak zdawać sobie sprawę, że czasem **mniejszej** wartości błędu odpowiada **gorsze** dopasowanie modelu



Walidacja krzyżowa

k-krotna walidacja krzyżowa:

Dane dzielone są na k podzbiorów rozłącznych, wykorzystywanych zarówno do uczenia jak i do szacowania dokładności modelu.

Następnie buduje się k modeli, używając każdej z części danych jako zbioru walidacyjnego, a pozostałych części jako zbioru uczącego.

Następnie dla każdego modelu oblicza się błąd i uśrednia go po wszystkich modelach, uzyskując tzw. błąd walidacji krzyżowej.

Dla oceny jakości modelu zbudowanego przez sieć neuronową stosuje się często metodę k-krotnej walidacji krzyżowej (ang. *k-fold cross-validation*).

Walidacja krzyżowa k-krotna (dla k=4)

Typowo stosuje się 5-krotną lub 10-krotną walidację krzyżowa lub N-krotną (N liczba wektorów uczących) zwaną wtedy „leave-one-out”

Bootstrap

Zakładamy zbiór uczący T składający się z P przykładów uczących $\mathbf{t}_p$ w postaci par uczących $(\mathbf{x}_p,\mathbf{y}_p)$

Bootstrap polega na tym, że losujemy (ze zwracaniem) spośród danych uczących $\mathbf{t}_i$ nowe P-elementowe zbiory i robimy to K-krotnie, tworząc w ten sposób K bootstrapowych zbiorów danych $T^{(k)}$.

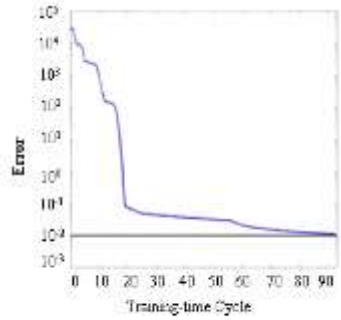
Na każdej z tych prób przeprowadzamy uczenie modelu neuronowego, wyznaczając interesujące wielkości, np. dokładność aproksymacji w określonych punktach E(T) otrzymując K estymat tych wyznaczonych wielkości.

Zagadnienia szczegółowe

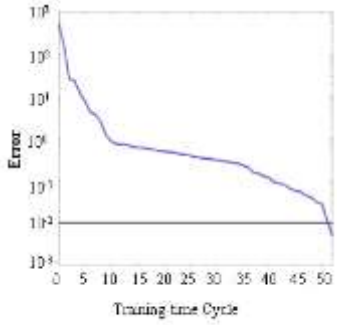
Czasami krzywe obrazujące zmiany błędu w trakcie uczenia mają nieco odmienny przebieg od wcześniej przedstawionego.

Pokażemy teraz kilka takich nietypowych przebiegów.

Czasem na typowej krzywej uczenia pojawiają się „przystanki” znamionujące konieczność przezwyciężania przez sieć pewnych kryzysów

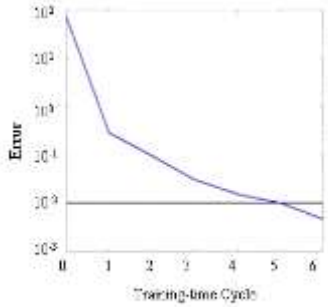


Niekiedy taki kryzys trwa dosyć długo, ale wytrwałym uczeniem można doprowadzić do ponownego znaczącego obniżenia wartości błędu

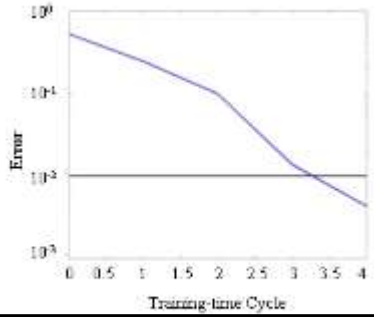


Niekiedy to końcowe przyspieszenie bywa wręcz drastyczne

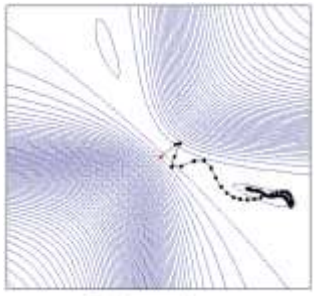
Czasem z kolei uczenie przebiega na tyle szybko, że przerywa się je po uzyskaniu zadowalającego stopnia dokładności mimo wciąż jeszcze widocznej możliwości dalszej poprawy wyników uczenia.



Czasem w zadaniach z dużą szybkością uczenia proces malenia błędu w późniejszych etapach zaczyna przyspieszać, zamiast zwolnić (co jest typowe)



Backpropagation jest generalnie metodą dosyć wolną w działaniu:

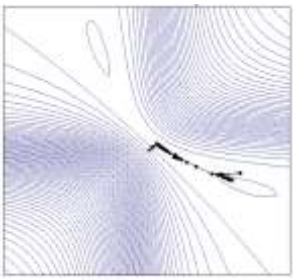


91 iterations

Niektórych badaczy to ogromnie denerwuje, więc na jednej konferencji naukowej rozprowadzane były takie ulotki:

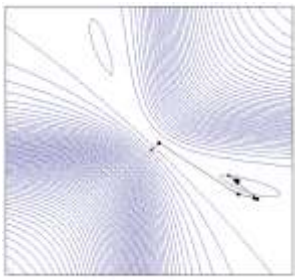
W wyniku tego niezadowolenia opracowano wiele metod uczenia, które w **łatwych** zadaniach są szybsze od *backpropagation*

Algorytm *Quickprop*



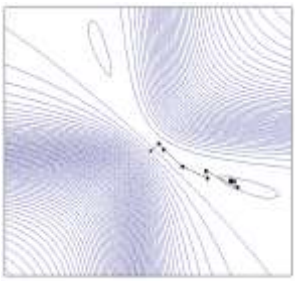
63 iterations

Metoda **gradientów sprzężonych**



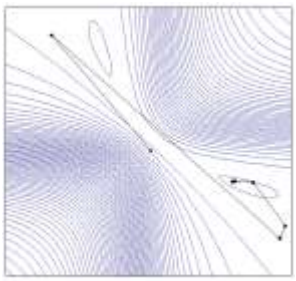
22 iterations

Metoda Newtona



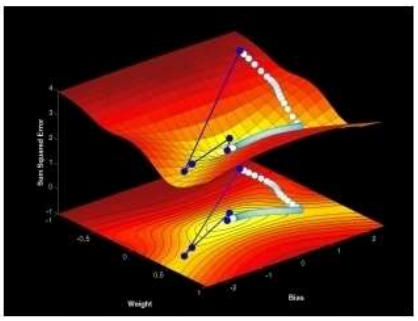
11 iterations

Algorytm Levenberga-Marquardta

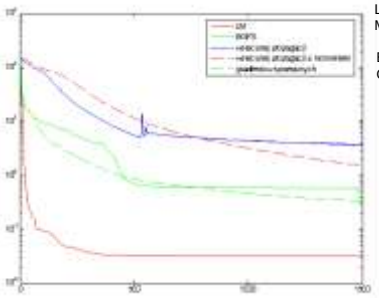


7 iterations

Porównanie drogi, jaką dochodzi do rozwiązania metoda *Levenberga-Marquardta* oraz *backpropagation*



Przebieg uczenia kilkoma metodami tej same sieci tego samego zadania



LM (Levenberga – Marquardta)

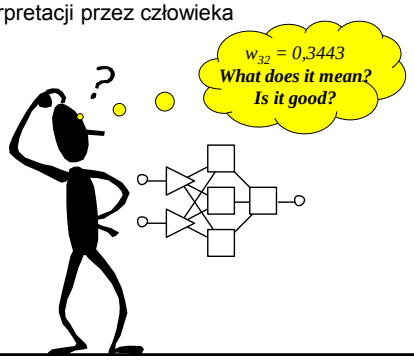
BFGS (Broyden-Fletcher-Goldfarb-Shanno)

Przykładowe porównanie jakości uczenia przy użyciu różnych algorytmów

Nr algorytmu	Nazwa algorytmu
1	Algorytm gradientowy
2	Algorytm gradientowy ze współczynnikiem momentum
3	Algorytm gradientowy ze współczynnikiem momentum i adaptacyjnie dobieranym współczynnikiem uczenia
4	Algorytm RPROP
5	Algorytm sprzężonych gradientów Fletcher-Reeves
6	Algorytm sprzężonych gradientów Polak-Ribiere
7	Algorytm sprzężonych gradientów Powell-Beale
8	Algorytm skalowanych sprzężonych gradientów
9	Algorytm one step secant
10	Algorytm BFGS
11	Algorytm Levenberg-Marquardt
12	Algorytm Bayesian regularization

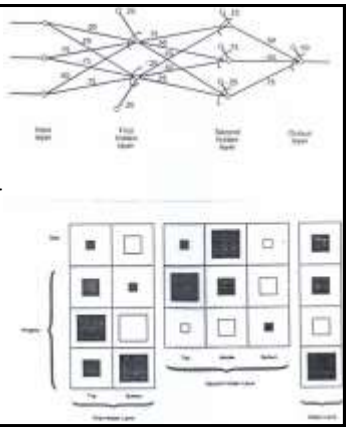
Niestety te bardziej wyrafinowane metody niekiedy fatalnie zawodzą w przypadku zadań naprawdę trudnych

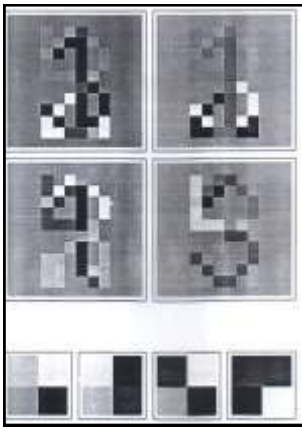
Wiedza, którą nabywa sieć neuronowa w trakcie uczenia jest zwykle niemożliwa do interpretacji przez człowieka



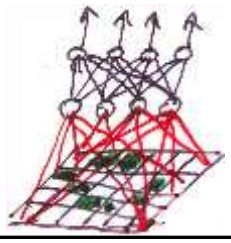
Czasem jednak udaje się „wyciągnąć” z sieci neuronowej zgromadzoną w niej wiedzę i nadać jej interpretowalną formę

Reprezentacja wiedzy w sieci neuronowej



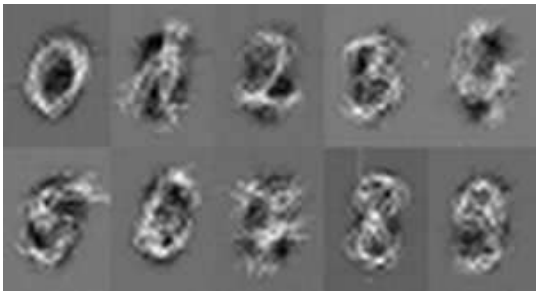


Rozkład wag odpowiednio w pierwszej i drugiej warstwie sieci uczonej rozpoznawania cyfr 1, 2, 3, 4

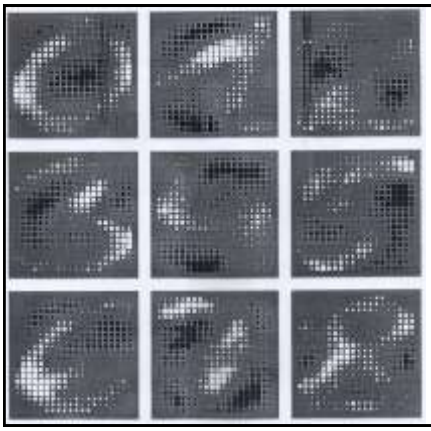


Rozkłady wag w sieci uczonej rozpoznawania ręcznie pisanych cyfr

Inny przykład tego samego zadania



Jeszcze dwa dalsze przykłady



Inny przykład rozkładu wag (dodatnich oraz ujemnych) po procesie uczenia rozpozna-wania cyfr

Koniec wykładu o uczeniu