

## Przyspieszanie uczenia: Bezwładność

Podobnie jak przy MADALINE przyspieszenie uczenia można uzyskać poprzez dodanie członu bezwładności do formuły zmiany wag:

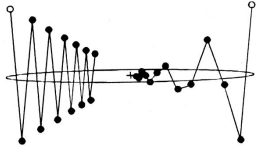
$$\Delta w_q^p(j+1) = -\eta \frac{\partial Q}{\partial w_q^p} + \alpha \Delta w_q^p(j)$$

$\alpha$  powinna być pomiędzy 0 a 1, często wybierana jest wartość 0.9.

Dla prawie płaskiej powierzchni kosztu

$$\Delta w_q^p \approx -\frac{\eta}{1-\alpha} \frac{\partial Q}{\partial w_q^p}$$

Chwilowe fluktuacje  $\frac{\partial Q}{\partial w_q^p}$  wpływają na  $\Delta w_q^p$  ze współczynnikiem  $\eta$  a długofalowe przyspieszenie jest rzędu  $\frac{1}{1-\alpha}$



$\eta = 0.0476$ , z lewej  $\alpha = 0$ , z prawej  $\alpha = 0.5$

## Przyspieszanie uczenia: Adaptacyjny dobór parametrów (1)

Ogólnie: sprawdza się czy określona modyfikacja wag rzeczywiście zmniejszyła funkcję kosztu  $Q$ . Jeśli nie, to trzeba zmniejszyć  $\eta$ .

Często stosowany schemat zmiany  $\eta$ :

$$\Delta \eta = \begin{cases} +a & \text{dla } \Delta Q < 0 & \text{systematycznie} \\ -b\eta & \text{dla } \Delta Q > 0 \\ 0 & \text{dla pozostałych przypadków} \end{cases}$$

W przypadku wykonania błędnego kroku, warto go wycofać.

## Przyspieszanie uczenia: Adaptacyjny dobór parametrów (2)

RESILIENT BACKPROPAGATION — do zmiany wag wykorzystujemy jedynie informację o znaku gradientu:

$$\Delta w_q^p(j) = \begin{cases} -\Delta_q^p(j) & \text{dla } \frac{\partial Q}{\partial w_q^p} > 0 \\ +\Delta_q^p(j) & \text{dla } \frac{\partial Q}{\partial w_q^p} < 0 \\ 0 & \text{dla pozostałych przypadków} \end{cases}$$

gdzie:  $Q$  jest miarą błędu po prezentacji wszystkich bodźców, zaś  $\Delta_q^p(j)$ :

$$\Delta_q^p(j) = \begin{cases} \eta^+ \Delta_q^p(j-1) & \text{jeżeli } \frac{\partial Q(j-1)}{\partial w_q^p} \cdot \frac{\partial Q(j)}{\partial w_q^p} > 0 \\ \eta^- \Delta_q^p(j-1) & \text{jeżeli } \frac{\partial Q(j-1)}{\partial w_q^p} \cdot \frac{\partial Q(j)}{\partial w_q^p} < 0 \\ \Delta_q^p(j-1) & \text{dla pozostałych przypadków} \end{cases}$$

gdzie  $0 < \eta^- < 1 < \eta^+$ .

Dodatkowo ustala się ograniczenie na możliwe wartości  $\Delta_{min} < \Delta_q^p < \Delta_{max}$ .

Standardowo  $\Delta_{min} \approx 10^{-6}$  a  $\Delta_{max} \approx 50$

Dla sigmoidalnych funkcji odpowiedzi metoda ta może poprawić uczenie w obszarze ogonów sigmoidy, gdzie wartość gradientu jest bardzo mała.

## Przyspieszanie uczenia: Inne procedury minimalizacji (1)

METODA NAJSZYBSZEGO SPADKU. Kolejnej wartości wagi szukamy wzdłuż prostej wyznaczonej przez poprzedni wektor wag  $\mathbf{w}^{(j)}$  i kierunek  $\mathbf{d}^{(j)}$ , zmieniając  $\lambda$  tak, aby zminimalizować  $Q$

$$\mathbf{w}^{(j+1)} = \mathbf{w}^{(j)} + \lambda \mathbf{d}^{(j)}$$

Kierunek  $d$  wybieramy przeciwnie do gradientu  $Q$

$$\mathbf{d}^{(j)} = -\nabla Q(\mathbf{w}^{(j)})$$

Zauważmy, że stary i nowy kierunek minimalizacji są ortogonalne:

$$0 = \frac{\partial}{\partial \lambda} Q(\mathbf{w}^{(j)} + \lambda \mathbf{d}^{(j)}) = \mathbf{d}^{(j)} \cdot \nabla Q(\mathbf{w}^{(j+1)})$$

## Przyspieszanie uczenia: Inne procedury minimalizacji (2)

METODA GRADIENTU SPRZEŻONEGO Poprzednią metodę można udoskonalić przez rezygnację z ortogonalności kolejnych kroków:

$$\mathbf{d}^{(j+1)} = -\nabla Q(\mathbf{w}^{(j+1)}) + \beta \mathbf{d}^{(j)}$$

i trzeba chytrze dobrać  $\beta$  tak, aby jak najmniej psuć efekt osiągnięty w poprzednim kroku. Nowy kierunek szukania powinien więc być taki, aby z dokładnością pierwszego rzędu nie zmienił składowej gradientu, która w poprzednim kroku została wyzerowana. A zatem chcemy, aby z dokładnością do wyrazów pierwszego stopnia, spełnione było:

$$\mathbf{d}^{(j)} \cdot \nabla Q(\mathbf{w}^{(j)} + \lambda \mathbf{d}^{(j+1)}) = 0$$

praktyczny sposób na znalezienie  $\beta$  spełniającego powyższy warunek podaje reguła Polaka-Ribiere'a:

$$\beta = \frac{(\nabla Q(\mathbf{w}^{(j+1)}) - \nabla Q(\mathbf{w}^{(j)})) \cdot \nabla Q(\mathbf{w}^{(j+1)})}{(\nabla Q(\mathbf{w}^{(j)}))^2}$$

## Przyspieszanie uczenia: Inne procedury minimalizacji (3)

METODY QUASI-NEWTONA oryginalna metoda Newtona polega na minimalizacji funkcji kosztu z wykorzystaniem drugich pochodnych.

Rozwijając  $Q(\mathbf{w})$  wokół bieżącego wektora wag  $\mathbf{w}_0$  mamy:

$$Q(\mathbf{w}) = Q(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0) \cdot \nabla Q(\mathbf{w}_0) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_0) \cdot H \cdot (\mathbf{w} - \mathbf{w}_0) + \dots \quad (*)$$

gdzie  $H_{ij} = \frac{\partial^2 Q}{\partial w_i \partial w_j}$

Różniczkowanie (\*) daje:

$$\nabla Q(\mathbf{w}) = \nabla Q(\mathbf{w}_0) + H \cdot (\mathbf{w} - \mathbf{w}_0) + \dots$$

chcemy znaleźć minimum  $Q$  czyli spełnić warunek  $\nabla Q(\mathbf{w}) = 0$ :

$$\nabla Q(\mathbf{w}_0) + H \cdot (\mathbf{w} - \mathbf{w}_0) = 0$$

stąd:

$$\mathbf{w} = \mathbf{w}_0 + H^{-1} \nabla Q(\mathbf{w}_0)$$

Wzór ten można stosować iteracyjnie.

Metoda w oryginalnej postaci jest bardzo kosztowna obliczeniowo ( $O(n^3)$ ) i jest niestabilna numerycznie. Stąd też realne implementacje są nieco inne i zasadniczo polegają na iteracyjnej aktualizacji hesjanu. Zwykle wymaga mniej kroków niż metoda gradientów sprzężonych, ale w każdym kroku jest więcej obliczeń i trzeba mieć pamięć na przechowywanie hesjanu.

## Przyspieszanie uczenia: Inne procedury minimalizacji (4)

LAVENBERG-MARQUARDT Metoda ta korzysta z faktu, że hesjan może być przybliżony przez:

$$H \approx J^T J$$

gdzie:  $J$  — macierz jacobiego

natomiast

$$\nabla Q(\mathbf{w}^{(j)}) = J^T (\mathbf{z}^{(j)} - \mathbf{y}^{(j)})$$

W metodzie tej wagi zmieniamy:

$$\mathbf{w}^{(j+1)} = \mathbf{w}^{(j)} + [J^T J + \mu I]^{-1} J^T (\mathbf{z}^{(j)} - \mathbf{y}^{(j)})$$

dla  $\mu = 0$  jest to metoda Newtona z przybliżoną wartością hesjanu, dla  $\mu$  dużego metoda dąży do zwykłej metody gradientowej. Metoda Newtona jest szybsza i dokładniejsza w pobliżu minimum  $Q$ .

$\mu$  jest zmniejszane po każdym udanym kroku a zwiększane jeśli w danym kroku  $Q$  wzrosło.

## Podsumowanie metod przyspieszania uczenia

**Ogólnie:** Za szybkość płacimy ilością koniecznej pamięci i wymaganą precyzją obliczeń

**Algorytm Lavenberg-Marquardt** jest najszybszym algorytmem w problemach aproksymacji funkcji dla sieci o średniej wielkości (do kilkuset wag). Prowadzi też, na ogół, w tych problemach do lepszego dopasowania w sensie błędu średniokwadratowego od pozostałych algorytmów. Jest jednak kosztowny jeśli chodzi o zapotrzebowanie na pamięć.

**Resilient Backpropagation** wydaje się być najlepszy w problemach rozpoznawania wzorców, ale nie nadaje się w zasadzie do aproksymacji funkcji. Nie jest pamięciożerny.

**Algorytm gradientów sprzężonych** jest najbardziej uniwersalnym algorytmem. Ma umiarkowane wymagania co do ilości pamięci.

**Zwykła metoda gradientowa** jest najwolniejsza, ale może to być użyteczne jeśli bardziej niż na czasie zależy nam na generalizacji.

## Minima lokalne

Wszystkie metody minimalizacji funkcji kosztu mogą utknąć w minimach lokalnych. Kilka metod, które mogą pomóc zmniejszyć problemy z minimami lokalnymi:

- uniknięcie wysycenia sigmoid już na samym początku uczenia. Np. dla unormowanego wejścia i dla sigmoidy z  $\beta = 1$  wybranie początkowych wag losowych o takich wartościach, że średnie pobudzenie neuronu  $e$  jest mniejsze, ale nie za bardzo niż 1 (można losować wagi rzędu  $\frac{1}{\sqrt{k_i}}$  gdzie  $k_i$  ilość wejść do jednostki  $i$ );
- poprawianie wag po każdej prezentacji wzorca, przy czym wzorce prezentowane są w losowej kolejności;
- zastosowanie jednostek stochastycznych — gradient oraz dodatkowy parametr  $T$  temperatura kontrolują prawdopodobieństwo zmiany wagi w określonym kierunku;
- delikatne losowe zmiany wag;
- przy każdej prezentacji wzorca dodawanie do niego troszkę szumu

Dodanie szumu zawsze spowolni proces uczenia, przy czym mała dawka może pomóc uniknąć minimów lokalnych, duża — znacznie spowalnia uczenie.

## Twierdzenie o potencjalnych możliwościach sieci

Sieć nieliniowa co najmniej dwuwarstwowa może aproksymować dowolną funkcję swoich wejść, ze z góry zadaną dokładnością.

Konieczna jest jedynie dostatecznie duża ilość jednostek w warstwach ukrytych.

Do aproksymacji dowolnej funkcji ciągłej wystarcza jedna warstwa ukryta.

Dowód nieformalny:

1. Każda “rozsądna” funkcja  $F_i\{X_k\}$  może być przedstawiona jako liniowa kombinacja “wypukłości”, z których każda jest różna od zera tylko w pobliżu  $X_k$ .
2. Takie “wypukłości” można skonstruować z dwóch warstw ukrytych.

Gdzie jest problem?

- twierdzenie mówi jedynie o istnieniu rozwiązania
- w ogólności nie wiadomo ile jednostek w warstwach stanowi “dostatecznie dużą ilość”
- nie ma gwarancji, że problem da się rozwiązać metodą wstecznej propagacji błędów.

## Reprezentacja danych — preprocessing

Przykład: Predykat dwu lub więcej grup

| Sygnal wejściowy pierwotny | Wynik | Permutacja sygnału wejściowego |
|----------------------------|-------|--------------------------------|
| - - - + + + - - -          | -1    | + - - + + - - -                |
| - + + - - + - - -          | +1    | + - - + - + - -                |
| + + + + + + + -            | -1    | + + - + + + + +                |
| + + - + + + - - -          | +1    | - - - + + + + +                |

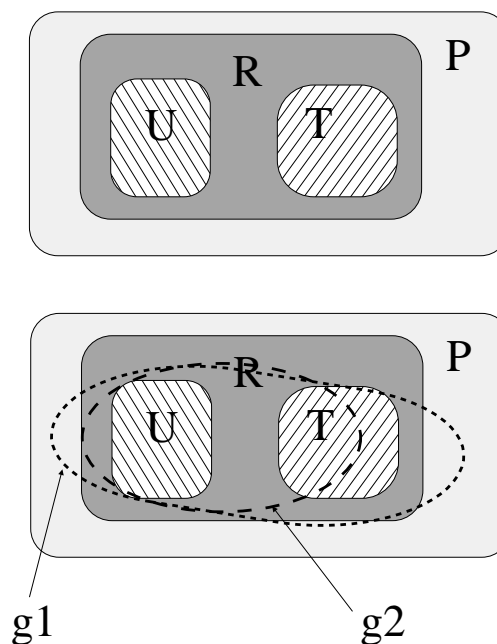
Dla sieci o połączeniach zupełnych między warstwami jest to bardzo trudne zadanie — sieć ta nie ma wbudowanej wiedzy o tym, że kolejność znaków jest istotną cechą.

Trywialne twierdzenie: uczenie sieci zawsze się uda jeśli zastosujemy prawidłowy preprocesor.

Jeśli w problemie występują symetrie, to o ile to możliwe warto przenieść ich analizę do fazy preprocessingu, bo powodują one powstanie w funkcji kosztu okresowości, wielokrotnych minimów lokalnych, płaskich dolin i wyżyn.

Ze standardowych technik przygotowywania danych (nie tylko dla sieci neuropodobnych) warto rozważyć: wyskalowanie danych, normalizację danych, przeprowadzenie analizy składowych głównych.

## Co to jest generalizacja?

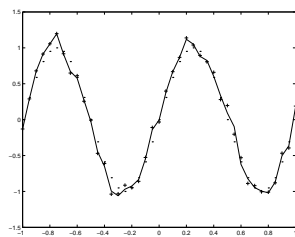


## Poprawianie generalizacji (1)

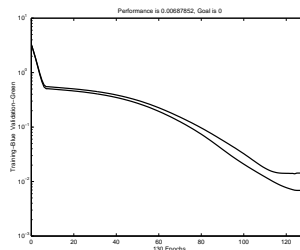
WCZESNE ZATRZYMYWANIE UCZENIA Zbiór dostępnych danych dzielimy na trzy części. Pierwszy podzbiór używamy do uczenia sieci. Drugi podzbiór jest używany do monitorowania błęd na nieznanym danych. Zazwyczaj w początkowej fazie uczenia następuje zmniejszanie błęd na zbiorze uczącym i na zbiorze monitorującym.

Oczekujemy, że w pewnym momencie błąd na zbiorze monitorującym zacznie rosnąć. Jeśli jest to efekt systematyczny to zatrzymujemy uczenie i wracamy do wag z minimum błęd na zbiorze monitorującym.

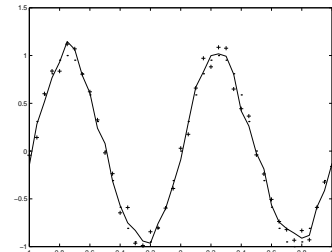
Trzeci podzbiór służy do testowania jakości generalizacji.



Bez stopu



Monitorowanie błędu



Z wczesnym stopem

## Optymalizacja architektury — obcinanie i zanikanie wag (1)

Pomysł: zaczynamy od dużej sieci i po pewnym cyklu uczenia przeanalizujemy połączenia w sieci usuwając mało istotne połączenia lub jednostki. Następnie powtórzmy uczenie.

Można spróbować tak zmodyfikować regułę zmiany wag, aby połączenia nieistotne same dążyły do 0;  
po standardowym kroku uczenia  
zmniejszamy wagi:

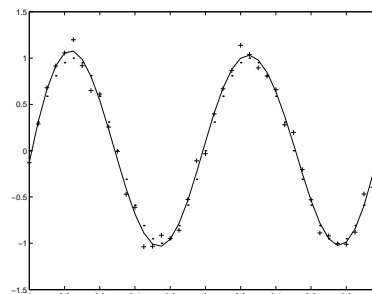
$$w_q^{p(j+1)} = (1 - \epsilon)w_q^{p(j)} \quad (*)$$

Jest to równoważne modyfikacji  $Q$ :

$$Q = Q_0 + \frac{1}{2}\gamma \sum_{pq} (w_q^p)^2$$

przy  $\epsilon = \eta\gamma$ . Konsekwencje:

- rozwiązanie jest "gładsze"
- wygładzenie funkcji kosztu likwiduje część minimów lokalnych



## Obcinanie i zanikanie wag (2)

Powyższa funkcja kosztu prowadzi do preferowania większej liczby małych wag zamiast jednej dużej. Sytuację poprawia wyrażenie:

$$Q = Q_0 + \frac{1}{2}\gamma \sum_{pq} \frac{(w_q^p)^2}{1 + (w_q^p)^2}$$

co jest równoważne (\*) przy  $\epsilon_q^p = \frac{\eta\gamma}{[1 + (w_q^p)^2]^2}$ . Dzięki temu małe wagi zanikają szybciej niż duże. To załatwia problem zanikania niepotrzebnych połączeń.

Aby zautomatyzować usuwanie zbędnych jednostek możemy zastosować:

$$\epsilon^p = \frac{\eta\gamma}{\left[1 + \sum_q (w_q^p)^2\right]^2}$$

dla wszystkich wejść do jednostki  $p$ .

Powoduje to szybsze zanikanie wag dla jednostek, które mają małe wagi wejściowe.

## Podsumowanie o sieciach jednokierunkowych z algorytmem wstecznej propagacji błędów

Sieci te nadają się do:

- aproksymacji odwzorowań przy zadanych jedynie danych wejście—wyjście
- klasyfikacji danych wejściowych

Porównanie z metodami klasycznymi — algorytmicznymi:

- aproksymacja — ostrożnie z generalizacją
- klasyfikacja sieciami vs. analiza dyskryminacyjna
- czarna skrzynka