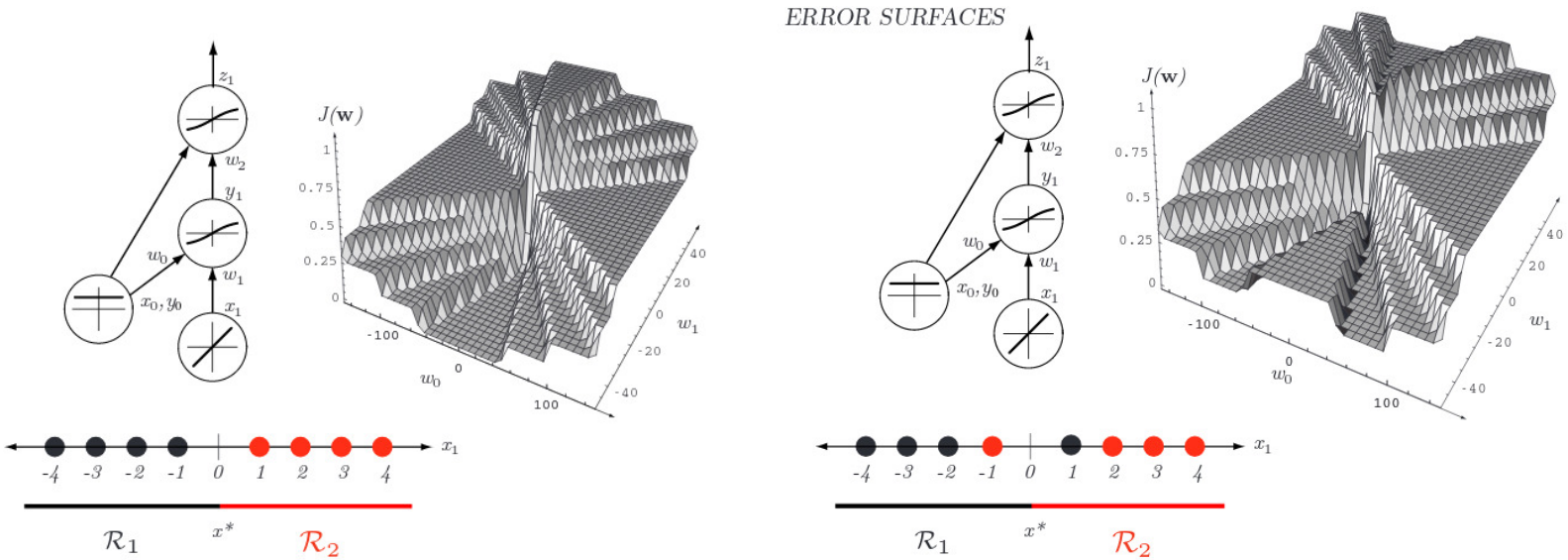
Ulepszenia treningu sieci MLP

Powierzchnia błędu



Minima lokalne i minimum globalne

Plateau - regiony o małej zmienności błędu względem wag

Jak omijać minima lokalne?

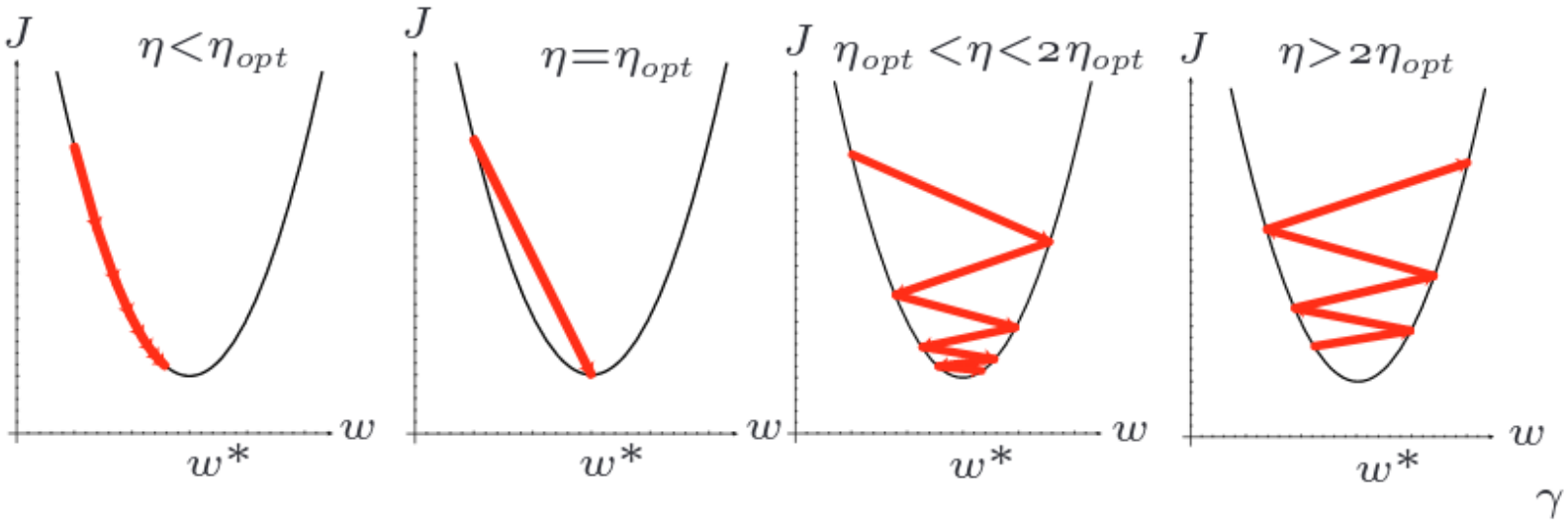
- Wielokrotny start z różnymi wartościami początkowymi - najprostsza ale skuteczna metoda
- Szum dodawany do wag lub szum dodany do danych pozwala wygładzić funkcję błędu i uciec z płytszych minimów – formalnie jest to równoważne regularyzacji, czyli dodaniu dodatkowego członu wygładzającego do funkcji błędu
- Losowa kolejność prezentowania przypadków
- Modyfikacje BP lub inne algorytmy optymalizacji

Metody globalnej minimalizacji

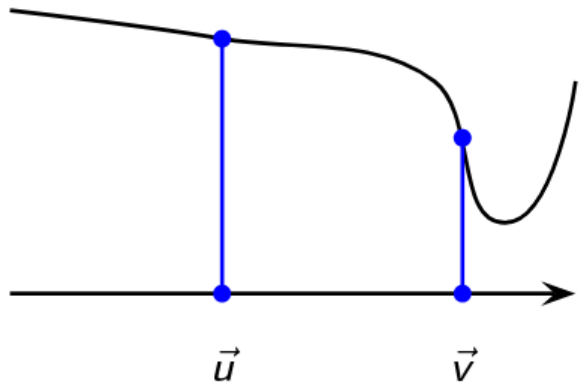
- Metody globalnej minimalizacji: wiele metod.
- Monte Carlo, symulowane wyżarzanie, metody multisympleksowe, minimalizacja Tabu, homotopia ...
- Dużo prac łączących algorytmy genetyczne z sieciami MLP
- Zalety: globalne, proste w realizacji, niektóre nie potrzebują gradientu, inne łączą zalety metod gradientowych z metodami globalnymi
- Wady: zwykle kosztowne i czasochłonne

Dobór kroku uczenia

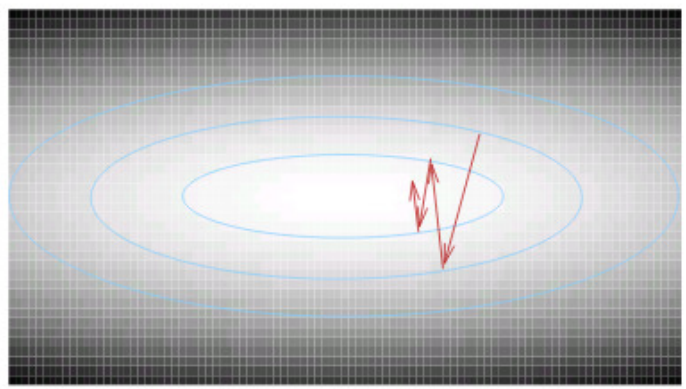
- zbyt mała wartość η - powolna zbieżność
- za duża wartość η - niestabilny trening, oscylacje
- wartość kroku może byc zmieniana w czasie uczenia, różne podejścia:
  - zmniejszana w czasie treningu
  - zwiększana dla płaskich powierzchni błędu
  - dobierana niezależnie do warstwy lub pojedynczych neuronów i wag



Płaskie powierzchnie i strome doliny



Powierzchnia błędu w wielu wymiarach ma różne nachylenie



Rys: Riedmiller, Machine Learning

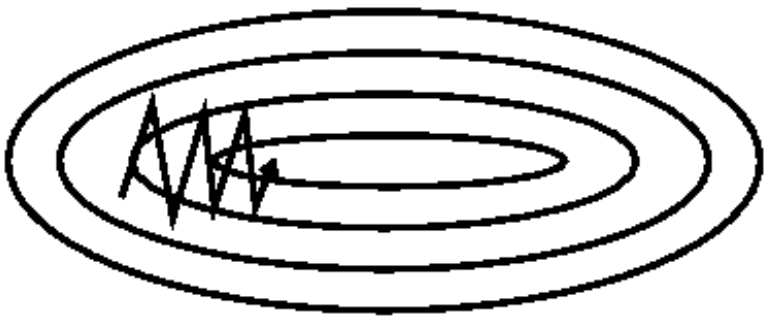
Trening z momentem

prędkość uczenia zależna od poprzednich wartości gradientów

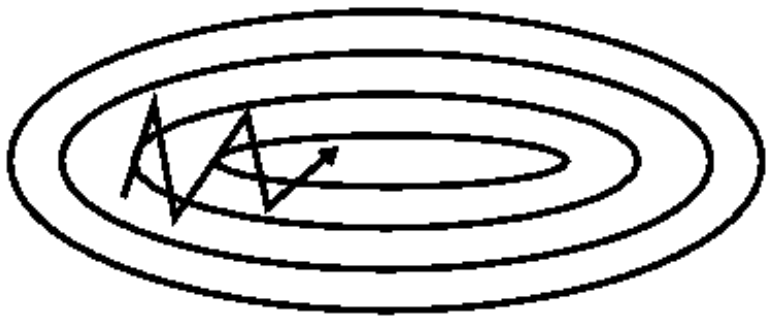
$$\Delta w(t) = -\eta \nabla E(t) + \gamma \Delta w(t-1)$$

- zwiększa krok uczenia, gdy gradient nie zmienia kierunku. Przyśpieszenie na płaskich odcinkach (gdzie ∇E stałe) wynosi w przybliżeniu
$$\Delta w(t) = -\frac{\eta}{1-\gamma} \nabla E$$
- redukuje oscylacje (spowalnia uczenie), gdy gradient zmienia kierunek
- gdy $\nabla E = 0$ wagi są nadal modyfikowane (bezwładność), może to ułatwić opuszczenie strefy „przyciągania" do minimum lokalnego
- $0 < \gamma < 1$, typowo współczynnik $\gamma = 0.9$

Trening z momentem



SGD bez momentu



SGD z momentem

Inicjalizacja wag

- Losowe wartości z rozkładu jednostajnego w okolicach 0
- Za duże wagi powodują duże wartości aktywacji i ryzyko nasycenia funkcji sigmoidalnych
- Dla d wejść można wybrać wartości z zakresu

$$-\frac{1}{\sqrt{d}} < w_{ij} < \frac{1}{\sqrt{d}}$$

  co przy standaryzacji danych daje średnio aktywację neuronów w zakresie liniowym $[-1, 1]$ sigmoidy
- Analogicznie dla kolejnych warstw

Skalowanie wartości wejściowych

- Rożne skale mierzonych cech x_i, „dzikie" rozkłady dalekie od rozkładu normalnego, wartości odstające
- Dane wejściowe $\mathbf{x}_i$ powinny posiadać zbliżone zakresy wartości
- Standaryzacja

$$x_s = \frac{x - \mu}{\sigma}$$

  μ - wartość średnia, σ - odchylenie standardowe
- Normalizacja w zakresie $[-1, +1]$

$$x_n = 2\frac{x - x_{min}}{x_{max} - x_{min}} - 1$$

RPROP

Resilent BP (Riedmiller, Braun, 1992)

- radzi sobie z problemem znikających (oraz za dużych) gradientów
- wyłącznie znak gradientu (nie amplituda) uwzględniany w obliczeniach
- stała uczenia dobierana dla każdej wagi niezależnie

$$\Delta w_{ij}(t) = -\eta_{ij}(t)\,\mathrm{sgn}\left(\frac{\partial E(\mathbf{w}(t))}{\partial w_{ij}}\right)$$

RPROP

- wymaga informacji z 2 ostatnich kroków uczenia
- η_{ij} rośnie, gdy brak zmiany kierunku gradientu
- η_{ij} maleje, gdy następuje zmiana kierunku

$$\Delta w_{ij}(t) = -\eta_{ij}(t)\,\mathrm{sgn}\left(\frac{\partial E(\mathbf{w}(t))}{\partial w_{ij}}\right)$$

gdzie

$$\eta_{ij}(t) = \begin{cases} \min(a \cdot \eta_{ij}(t-1), \eta_{max}) & \text{dla} \quad \frac{\partial E(\mathbf{w}(t))}{\partial w_{ij}} \frac{\partial E(\mathbf{w}(t-1))}{\partial w_{ij}} > 0 \\ \max(b \cdot \eta_{ij}(t-1), \eta_{min}) & \text{dla} \quad \frac{\partial E(\mathbf{w}(t))}{\partial w_{ij}} \frac{\partial E(\mathbf{w}(t-1))}{\partial w_{ij}} < 0 \\ \eta_{ij}(t-1) & \text{w pozostałych przypadkach} \end{cases}$$

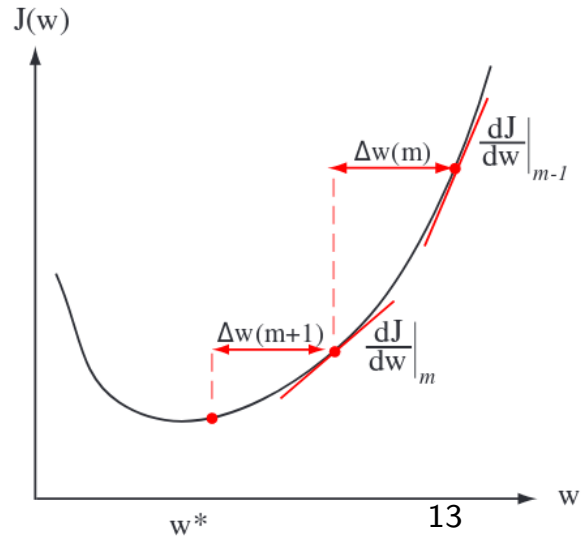
$$a = 1.2, \quad b = 0.5, \quad \eta_{min} = 10^{-6}, \quad \eta_{max} = 50$$

Quickprop (Fahlman, 1988)

- założenie: wagi są niezależne a funkcja błędu w okolicach minimum może być przybliżona parabolą
- przybliżenie za pomocą wartości i gradientów funkcji w 2 punktach $w(m)$ i $w(m-1)$

$$\Delta w(m+1) = \frac{\nabla_{ij} E(m)}{\nabla_{ij} E(m-1) - \nabla_{ij} E(m)} \Delta w(m)$$

- wagi mają niezależną zbieżność uczenia
- zbieżność kwadratowa
- trening może być niestabilny



Metody 2 rzędu

Rozwinięcie w szereg Taylora:

$$E(\mathbf{w}+\Delta\mathbf{w}) \approx E(\mathbf{w}) + \nabla E(\mathbf{w})^T \Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^T \nabla^2 E(\mathbf{w}) \Delta\mathbf{w}$$

gdzie $\nabla^2 E(\mathbf{w}) = \mathbf{H}$ jest macierzą $n \times n$ pochodnych 2 rzędu (Hessian)

$$H_{ij} = \frac{\partial^2 E(\mathbf{x};\mathbf{w})}{\partial w_i \partial w_j}$$

Gradient funkcji kosztu:

$$\nabla E(\mathbf{w}+\Delta\mathbf{w})^T \approx \nabla E(\mathbf{w})^T + \Delta\mathbf{w}^T \mathbf{H}$$

Minimum osiągane dla $\nabla E(\mathbf{w}+\Delta\mathbf{w}) = 0$, stąd

$$\Delta\mathbf{w} = -\mathbf{H}^{-1} \nabla E(\mathbf{w})$$

rozwiązanie w jednym kroku, jeżeli potrafimy obliczyć $\mathbf{H}^{-1}$

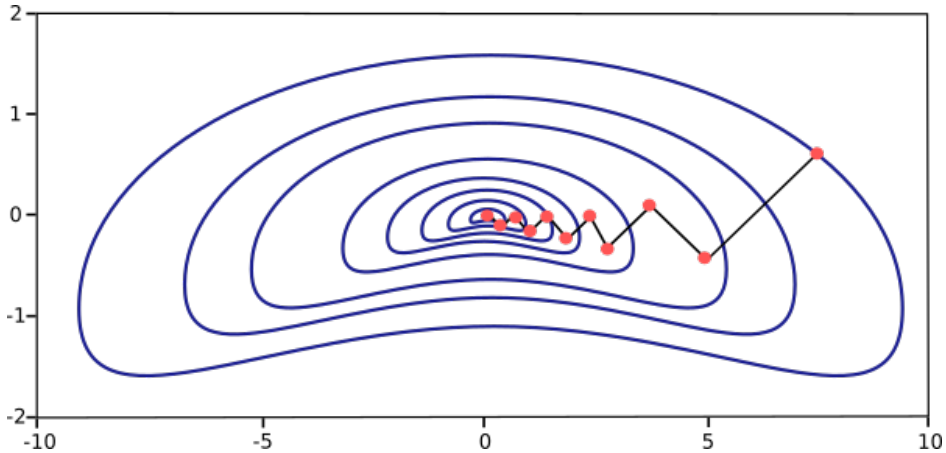
Metoda Newtona

Iteracyjna metoda 2 rzędu:

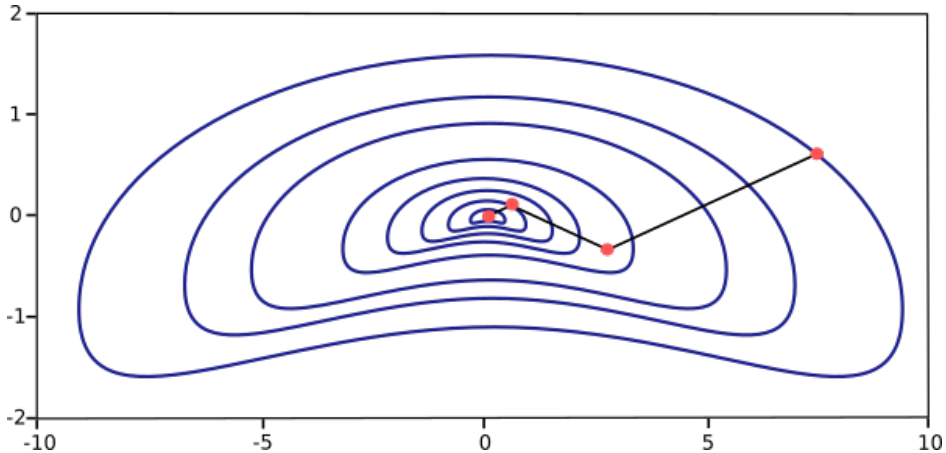
$$\mathbf{w}(t+1) = \mathbf{w}(t) - \mathbf{H}^{-1}\nabla E(\mathbf{w}(t))$$

- metoda kosztowna czasowo $O(n^3)$ (odwracanie macierzy w każdej iteracji)
- zbieżność (kwadratowa) po kilku iteracjach
- metoda kosztowna pamięciowo, Hessian $O(n^2)$, gdzie n liczba wag
- praktyczne zastosowania tylko dla małych sieci
- $\mathbf{H}$ nie zawsze jest dodatnio określona - stosuje się aproksymacje

Spadek gradientu



Metoda Newtona



Rys: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network

Metody quasi-Newtona

Przybliżenia do Hesjanu

- zaniedbanie pozadiagonalnych elementów

$$\Delta w_i = -\frac{\partial E}{\partial w_i} \Big/ \frac{\partial^2 E}{\partial w_i^2}$$

- metoda zmiennej metryki - przybliżenie do H^{-1} oraz iteracyjna metoda Newtona, kwadratowo zbieżna
  - Davidon-Fletcher-Power (DFP)
  - Broyden-Fletcher-Goldfarb-Shanno (BFGS).
- Metoda Levenberg-Marquardta oparta jest na przybliżeniu Gaussa-Newtona

Metoda Levenberg-Marquardta

Metoda LM znajduje rozwiązanie zadań optymalizacji, które daje się sprowadzić do postaci

$$E = \sum_{i=1}^{p} e_i^2$$

W przypadku danych zawierających n przypadków i sieci o m wyjściach sumowanie obejmuje $n \cdot m$ wartości dla funkcji błędu MSE:

$$E = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (f_j(\mathbf{x}_i) - y_{ij})^2$$

Niech $\mathbf{e} = [e_1, e_2, \ldots, e_p]$ tworzy **wektor rezydualny** zaś **J** macierz **Jakobianu**

$$J_{ij} = \frac{\partial e_i}{\partial w_j}, \qquad i = 1, \ldots, p \qquad j = 1, \ldots, k$$

gdzie k to ilość wszystkich wag.

Metoda Levenberg-Marquardta

Gradient funkcji błędu można wyrazić w postaci:

$$\nabla E = 2\mathbf{J}^T\mathbf{e}$$

Metoda LM zastępuje macierz Hessego w metodzie Newtona przybliżeniem opartym na wartości grandiemu wraz z czynnikiem regularyzacyjnym μ

$$\mathbf{H} \approx 2\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}$$

Współczynnik tłumienia μ zapewnia dodatnią określoność $\mathbf{H}$.

Reguła aktualizacji wag:

$$\Delta\mathbf{w} = -2\left(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right)^{-1}\mathbf{J}^T\mathbf{e}$$

Metoda Levenberg-Marquardta

$$\Delta \mathbf{w} = -2 \left(\mathbf{J}^T \mathbf{J} + \mu \mathbf{I} \right)^{-1} \mathbf{J}^T \mathbf{e}$$

- dla dużych μ czynnik kwadratowy zanika i uzyskujemy metodę największego spadku
- dla $\mu = 0$ mamy metodę Newtona o kwadratowej zbieżności
- LM startuje z dużą wartością μ a następnie w trakcie uczenia w miarę zbliżania się do rozwiązania μ jest zmniejszane
- bardzo szybko zbieżna metoda dla funkcji, które są sumą kwadratów (MSE)
- nie nadaje się do zastosowań z funkcją Cross Entropy
- nie praktyczne dla dużych danych, duży koszt pamięci, Jakobian ma wymiar $p \times k$, gdzie $p = n \cdot m$ - ilość wektorów razy ilość wyjść, k - ilość parametrów (wag)

Metoda Levenberg-Marquardta

Algorytm doboru współczynnika μ w i-tym kroku, dla danego $\mathbf{w}(i)$:

1. oblicz wartość $\mathbf{w}(i+1)$ zgodnie z regułą uczenia LM
2. oblicz wartość błędu w punkcie $\mathbf{w}(i+1)$
3. jeśli błąd wzrósł, wróć do poprzedniej wartości $\mathbf{w}(i)$ i zwiększ wartość μ k-krotnie i wróć do kroku 1 (przybliżenie liniowe minimalizowanej funkcji w otoczeniu $\mathbf{w}(i)$ okazało się nie dość ścisłe, więc zwiększamy „wpływ" metody największego spadku)
4. jeśli błąd zmalał, zaakceptuj ten krok i zmniejsz wartość μ k-krotnie (założenie o liniowości minimalizowanej funkcji w otoczeniu $\mathbf{w}(i)$ okazało się wystarczająco ścisłe, więc zwiększamy „wpływ" metody Gaussa-Newtona).

Typowo $k = 10$.

Metoda gradientów sprzężonych

Metoda **gradientów sprzężonych** (*conjugated gradients*) poszukuje minimum wzdłuż kierunku $\Delta\mathbf{w}(m)$ ortogonalnego i sprzężonego do wszystkich kierunków z poprzednich iteracji.

Kierunki $\Delta\mathbf{w}(m)$ i $\Delta\mathbf{w}(m-1)$ są sprzężone gdy:

$$\Delta\mathbf{w}^T(m-1)\mathbf{H}\Delta\mathbf{w}(m) = 0$$

Kierunek wyszukiwania minimum w iteracji m wyznaczamy z reguły

$$\Delta\mathbf{w}(m) = -\nabla E(m) + \beta(m)\Delta\mathbf{w}(m-1)$$

gdzie **współczynnik sprzężenia** β kumuluje informacje o kierunkach z poprzednich iteracji.

Metoda gradientów sprzężonych

Przykładowe postaci współczynnika sprzężenia:

- reguła Fletchera-Reevesa

$$\beta(m) = \frac{\nabla E(m)^T \nabla E(m)}{\nabla E(m-1)^T \nabla E(m-1)}$$

- reguła Polaka-Ribiera

$$\beta(m) = \frac{\nabla E(m)^T (\nabla E(m) - \nabla E(m-1))}{\nabla E(m-1)^T \nabla E(m-1)}$$

- reguła Hestenesa-Stiefela

$$\beta(m) = \frac{\nabla E(m)^T (\nabla E(m) - \nabla E(m-1))}{-\nabla E(m-1)^T (\nabla E(m) - \nabla E(m-1))}$$

- reguła Dai–Yuan

$$\beta(m) = \frac{\nabla E(m)^T \nabla E(m)}{\nabla E(m-1)^T \nabla E(m-1)}$$

Algorytm metody gradientów sprzężonych

Pierwszy kierunek

1. zainicjuj $\mathbf{w}(0)$ i znajdź pierwszy kierunek $\Delta\mathbf{w}(0)$ metodą największego spadku $\Delta\mathbf{w}(0) = -\nabla E(0)$
2. wykonaj liniowe przeszukiwanie $\alpha = \arg\min_\alpha E(\mathbf{w}(m) + \alpha\Delta\mathbf{w}(m))$
3. zaktualizuj wagi $\mathbf{w}(1) = \mathbf{w}(0) + \alpha\Delta\mathbf{w}(0)$

Kolejne kierunki

1. wyznacz kierunek $\Delta\mathbf{w}(m) = -\nabla E(m)$ metodą największego spadku
2. wyznacz współczynnik sprzężenia $\beta(m)$
3. zaktualizuj kierunek sprzężony $\Delta\mathbf{w}(m) = -\nabla E(m) + \beta(m)\Delta\mathbf{w}(m-1)$
4. wykonaj liniowe przeszukiwanie $\alpha = \arg\min_\alpha E(\mathbf{w}(m) + \alpha\Delta\mathbf{w}(m))$
5. zaktualizuj wagi $\mathbf{w}(m+1) = \mathbf{w}(m) + \alpha\Delta\mathbf{w}(m)$

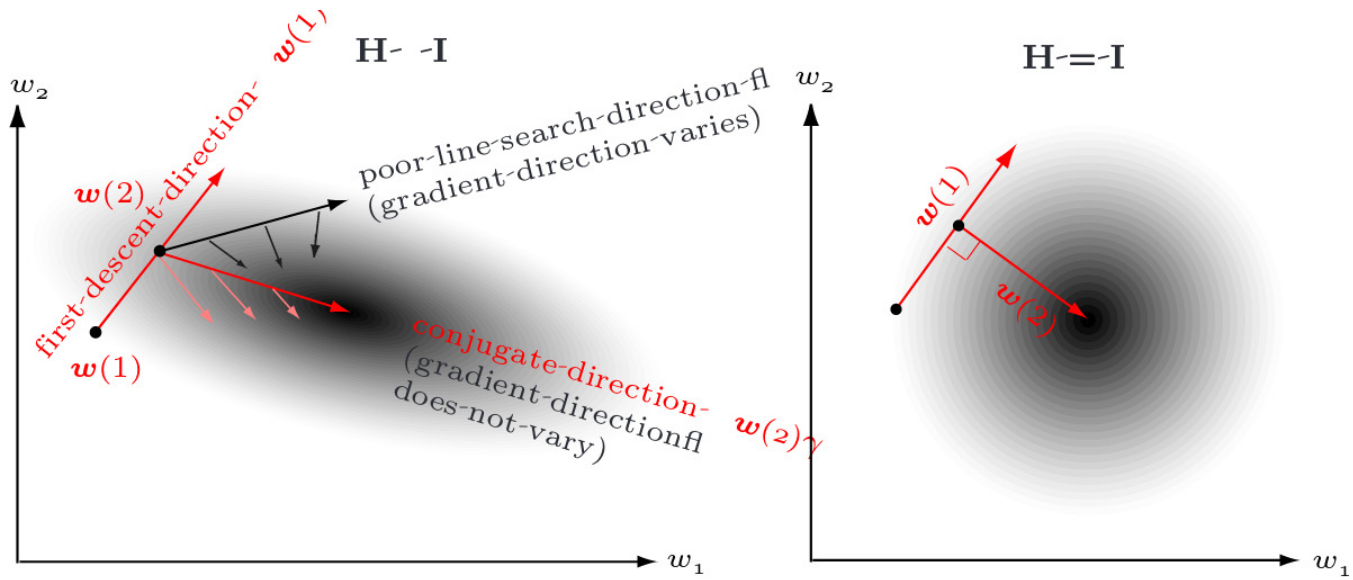
Metoda gradientów sprzężonych

- ponieważ na skutek przybliżeń w kolejnych iteracjach kierunki mogą zatracić ortogonalność po n krokach (gdzie n jest liczbą optymalizowanych parametrów) metodę inicjujemy ponownie w ostatnio znalezionym punkcie
- dla kwadratowej funkcji kosztu w n wymiarach metoda CG osiąga minimum w n krokach
- zbieżność liniowa lecz znacznie szybsza od GD bez konieczności wyznaczania odwrotności hesjanu $\mathbf{H}^{-1}$
- małe zapotrzebowanie pamięciowe, stosunkowo mała złożoność obliczeniowa pozwala stosować do bardzo złożonych problemów optymalizacyjnych

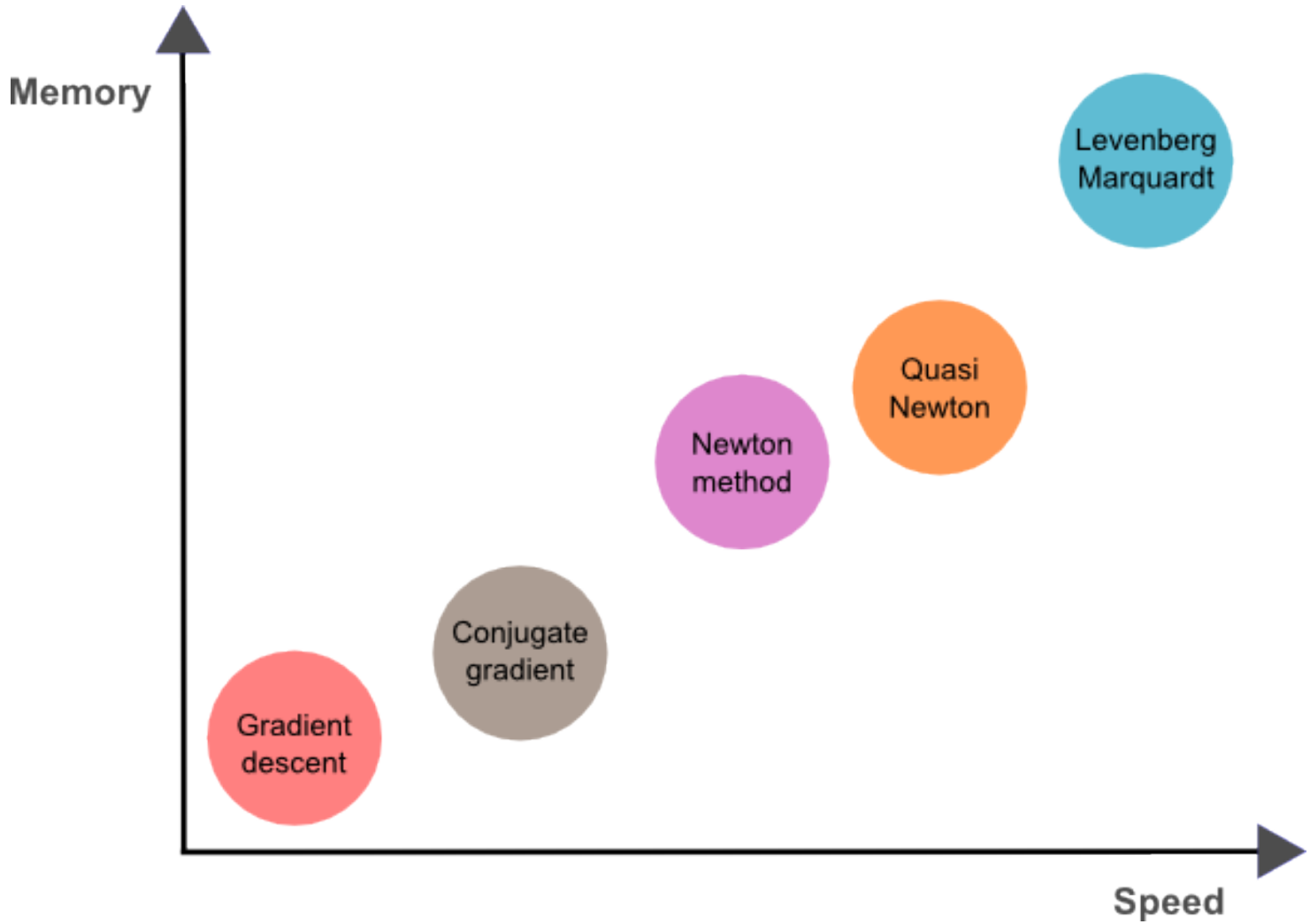
https://en.wikipedia.org/wiki/Nonlinear_conjugate_gradient_method

Minimalizacja CG

- kolejny kierunek sprzężony nie wpływa ujemnie na wartość błędu wzdłuż poprzednio wyszukanych kierunków
- dla **H** diagonalnej kolejne kierunki są ortogonalne względem siebie



Duda, Hart, Pattern recognition



Rys: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network