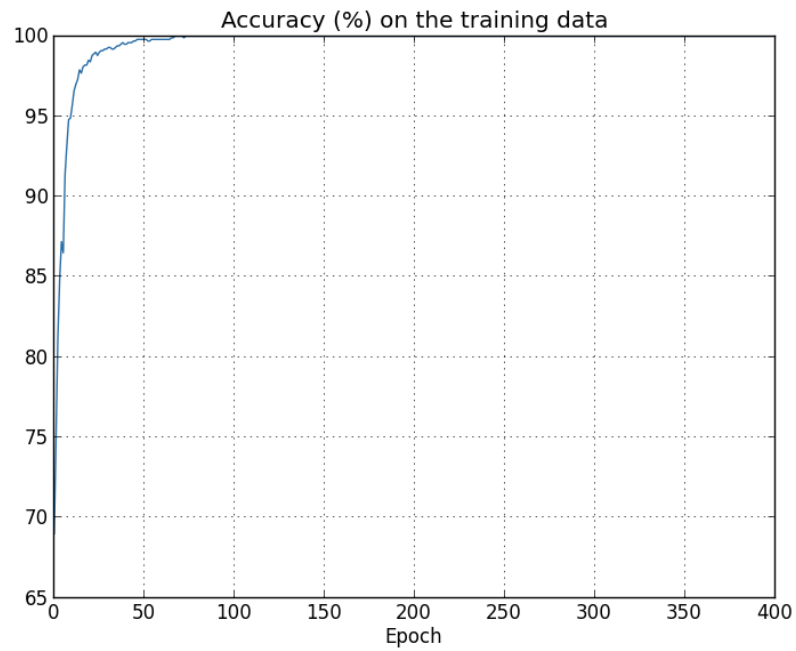
Generalizacja i regularyzacja sieci MLP

Generalizacja

Wartość funkcji błędu dla zbioru treningowego może osiągnąć 0.
Czy to znaczy, że mamy najlepszy model?



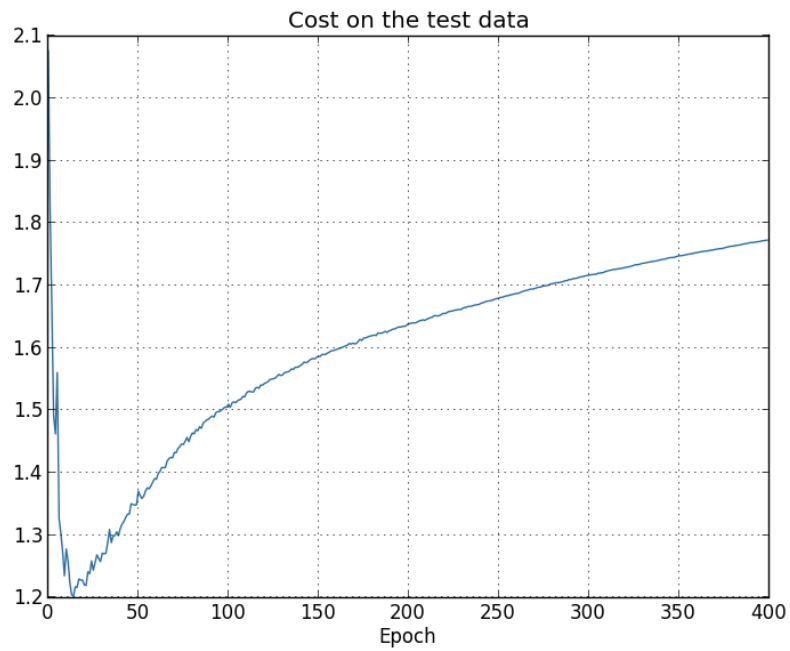


Rys: http://neuralnetworksanddeeplearning.com/

Generalizacja

Wartość funkcji błędu dla zbioru treningowego może osiągnąć 0.
Czy to znaczy, że mamy najlepszy model?

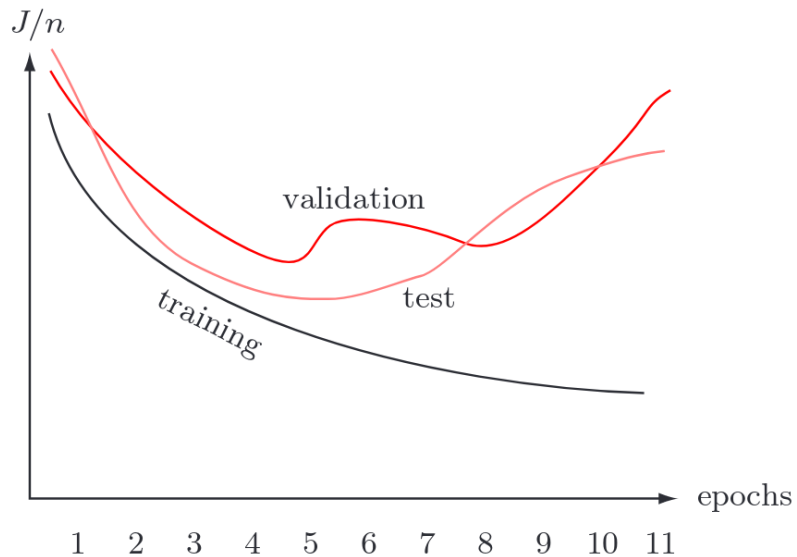




Rys: http://neuralnetworksanddeeplearning.com/

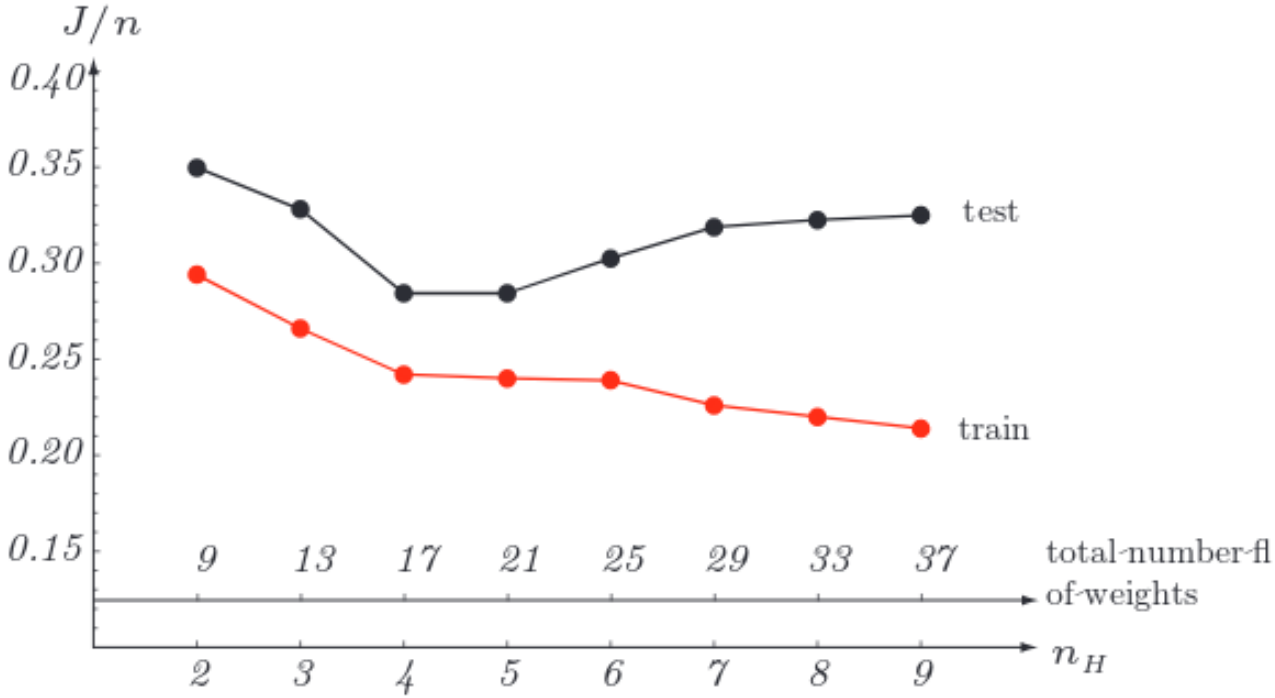
Generalizacja

- Celem uczenia jest **generalizacja**, zdolność modelu do uogólniania wiedzy pozwalającej przewidywać wyniki dla nowych, nieznanych danych
- W praktyce dążymy do minimalizacji błędu na danych, które nie zostały użyte w treningu (tzw. **zbiór testowy**)
- Zbiór **walidacyjny** to wyodrębniony fragment danych (najczęściej ze zbioru uczącego) pozwalający oszacować błąd generalizacji w trakcie procesu uczenia.
  Jest przydatny przy określaniu punktu zatrzymania treningu.



Duda, Hart, Pattern Recognition

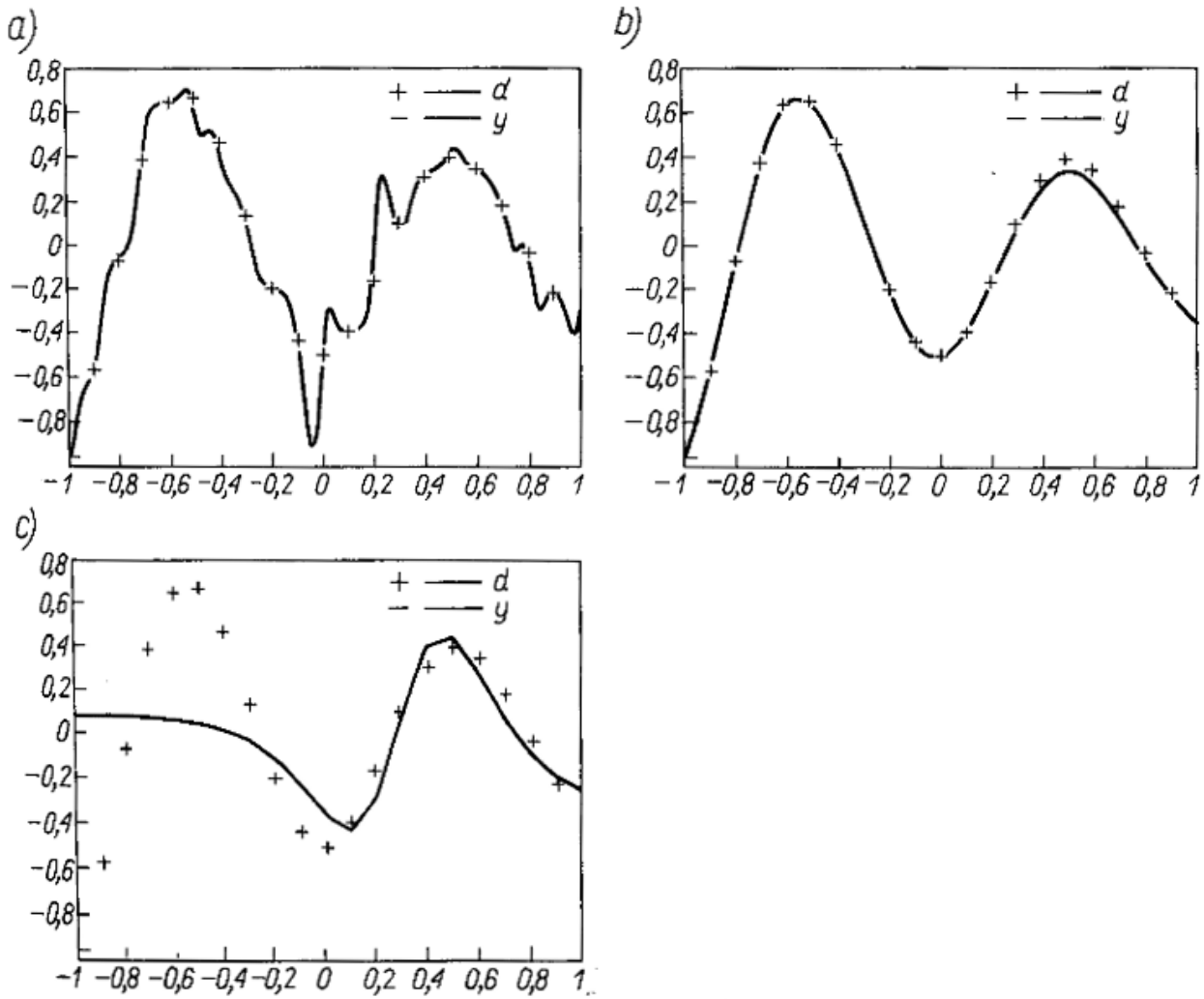
Rozmiar sieci



- liczba wag określa liczbę stopni swobody - liczba próbek uczących powinna być zwykle wielokrotnością liczby parametrów, (np. 10-krotnością liczby wag)
- za dużo wag $\rightarrow$ złożony model „uczy się na pamięć” (przeuczenie)
- za mało wag $\rightarrow$ model zbyt prosty w stosunku do problemu (niedouczony)

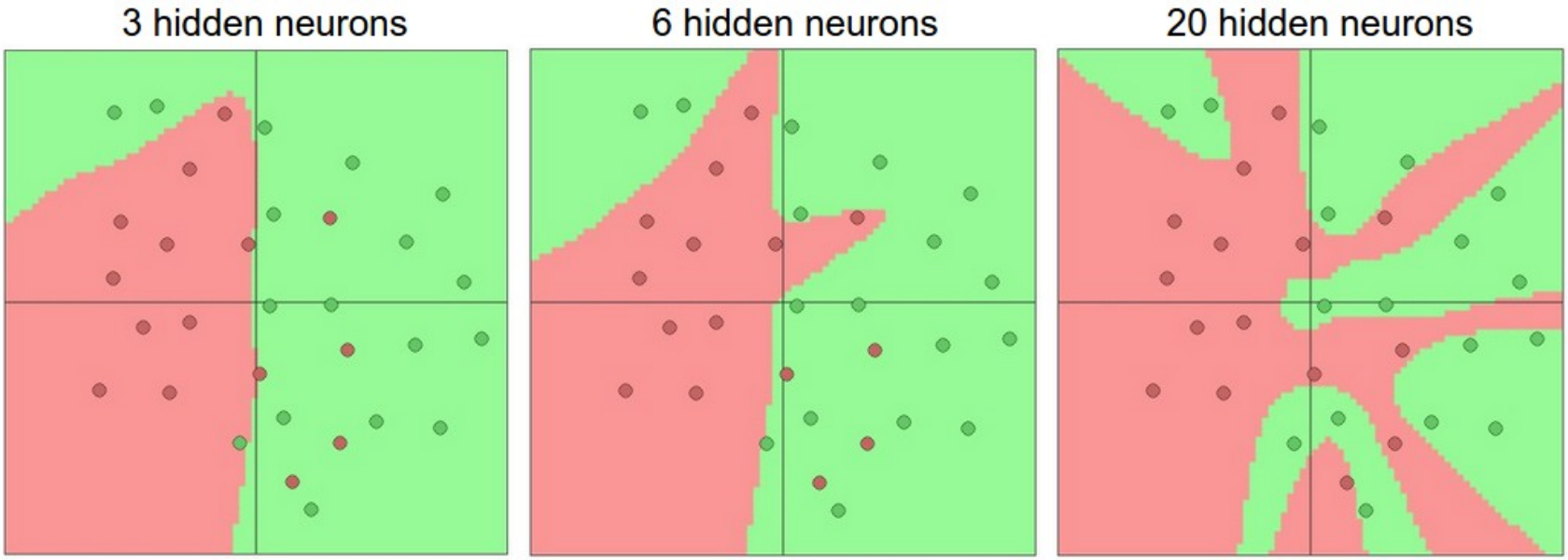
Rozmiar warstwy ukrytej

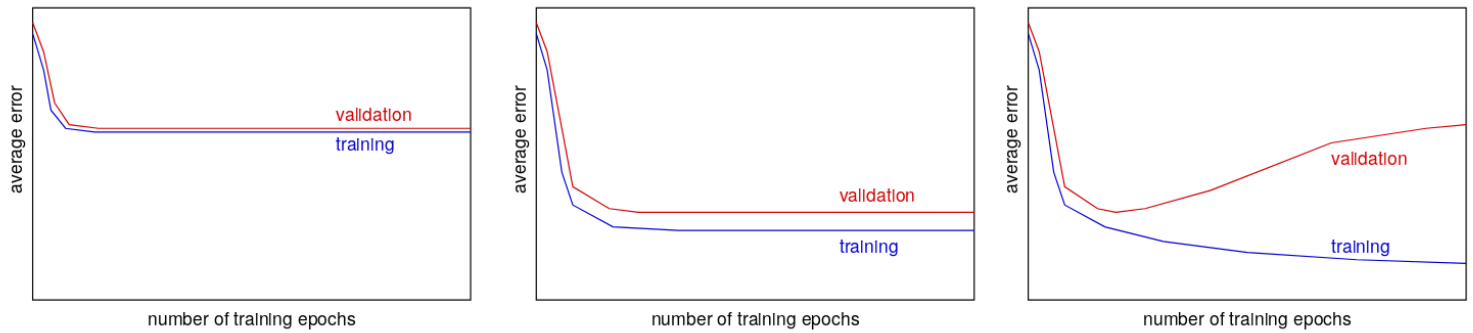
Zdolność uogólniania w problemie aproksymacji



Rozmiar warstwy ukrytej

Zdolność uogólniania w problemie klasyfikacji





- niedouczenie - błąd treningowy i walidacyjny pozostają duże
- przeuczenie - błąd walidacyjny rośnie a błąd treningowy maleje

Rys: Riedmiller, Machine Learning

Metody regularyzacji sieci neuronowych

Regularyzacja to metody, których celem jest poprawienie generalizacji

- dobór wielkości modelu (ilości neuronów i wag), preferowane są najprostsze rozwiązania (brzytwa Ockhama)
  - spośród wielu modeli wybierz ten o najmniejszym błędzie walidacyjnym i najmniejszej liczbie parametrów (neuronów)
  - **pruning** - usuwanie nadmiarowych połączeń lub neuronów z wytrenowanej sieci
  - metody **konstrukcyjne** (rosnące) - zacznij od najprostszego modelu i zwiększaj jego złożoność w kolejnych iteracjach
  - sieci **ontogeniczne** - rozrastające się i kurczące się w czasie treningu
- wymuszanie małych absolutnych wartości wag lub ich zanikanie
  - wagi zerowe oznaczają brak połączenia
  - neurony sigmoidalne z małymi wagami są w przybliżeniu liniowe

Regularyzacja

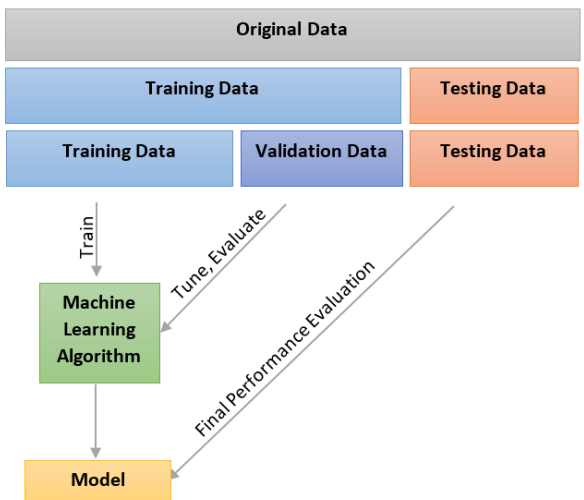
- modyfikacje funkcji kosztu wymuszające odpowiednią strukturę sieci, np. wymuszające minimalizację wartości wag

$$\bar{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

  gdzie $\lambda > 0$ steruje wielkością regularyzacji
- **wczesne zatrzymanie** (*early stopping*) - strategia polegająca na przerwaniu procesu uczenia zanim wystąpi przeuczenie (np. gdy błąd walidacyjny zacznie rosnąć)
- zwiększenie rozmiaru zbioru uczącego - więcej danych pozwala na lepsze dopasowanie modelu do problemu
- przetransformowanie danych do takiej postaci, która zwiększy szansę znalezienie optymalnego rozwiązania (ekstrakcja cech, normalizacja, itp.)
- selekcja cech - wybór tylko istotnych zmiennych do treningu
- dodanie szumu do wag lub do danych uczących
- uśrednianie wyników z wielu modeli, np.: boosting, bagging

Trening i walidacja

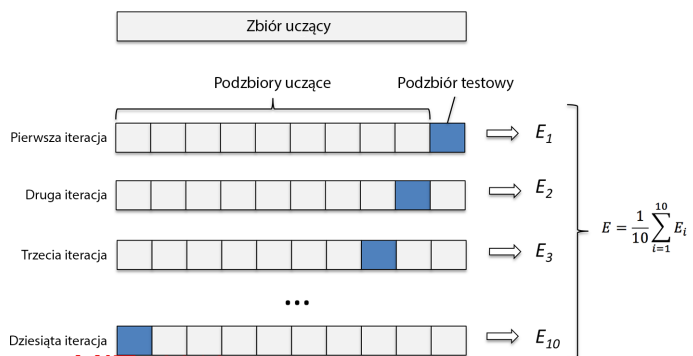
- uczenie sieci minimalizuje błąd treningowy a nie błąd generalizacji
- **zbiór walidacyjny** pozwala oszacować błąd generalizacji w trakcie treningu. Wydzielany ze zbioru treningowego, więc powoduje zmniejszenie liczby dostępnych próbek uczących.
- **zbiór testowy** używany wyłącznie do ewaluacji nauczonego modelu



Rys. S. Mostafa Eissa, ☞ Introduction to Machine Learning , 2016

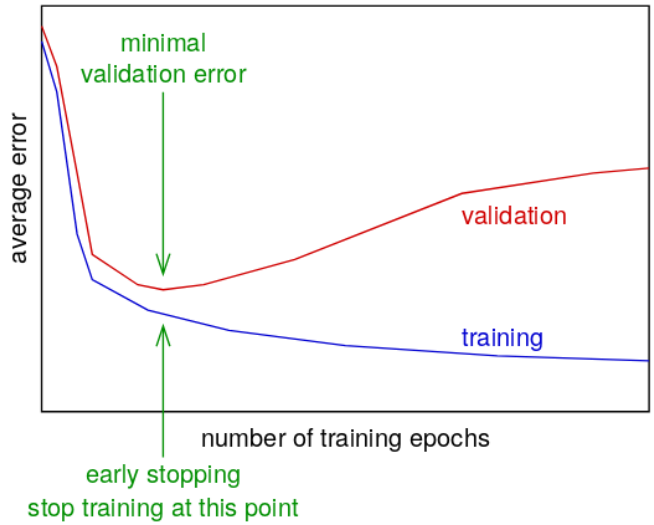
Walidacja krzyżowa (kroswalidacja)

- **Kroswalidacja** (walidacja krzyżowa) - metoda pozwalająca na oszacowanie błędu generalizacji i wariancji modelu poprzez powtarzanie treningu i ewaluacji na kolejnych podziałach zbioru uczącego
- k-krotna kroswalidacja - dzieli zbiór na k części, gdzie $k-1$ części jest używanych do treningu a reszta do testu
- kroswalidacja **leave-one-out** (LOO) - rodzaj kroswalidacji, gdzie N jest równe liczbie wektorów uczących, tj. trening odbywa się na $N-1$ przypadkach, predykcja dotyczy pojedynczego przypadku. Stosowane dla bardzo małych danych.
- średni błąd wyznaczany jest z k treningów (foldów)



Rys. S. Raschka, Python Machine Learning, MIT, 2019

Wczesne zatrzymanie



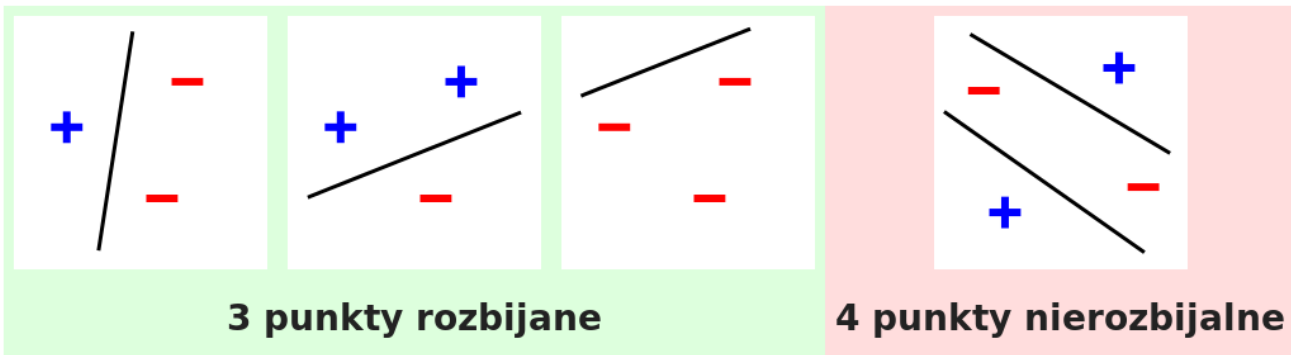
Strategie:

- przerwij trening w momencie, gdy błąd walidacyjny zacznie rosnąć (np. gdy brak poprawy przez k epok). Często wystarczy tylko kilka iteracji.
- trenuj przez N epok i zapamiętaj model o najmniejszym błędzie walidacyjnym wyznaczonym po każdej epoce

Rys: Riedmiller, Machine Learning

Wymiar Vapnika-Chervonenkisa

Wymiar $VC_{dim}(\mathbb{H})$ w klasie hipotez $\mathbb{H}$ to największa liczba przypadków uczących, którą $\mathbb{H}$ jest w stanie *rozbić*, tj. podzielić przy dowolnej kombinacji etykiet (2^N możliwości w przypadku 2 klas)



Dla perceptronu z 2 wejściami $y = w_1 x_1 + w_2 x_2 + b$

$$\mathrm{VC}_{dim}(\mathrm{Perceptron}(\mathbb{R}^2)) = 3$$

W przestrzeni $\mathbb{R}^d$ mamy $(d+1)$ parametrów stąd

$$\mathrm{VC}_{dim}(\mathit{Perceptron}(\mathbb{R}^d)) = d+1$$

Twierdzenie Vapnika-Chervonenkisa (Vapnik, 1995)

Błąd generalizacji $E_G(h)$ hipotezy h pochodzącej z klasy hipotez $\mathbb{H}$ o skończonym wymiarze VC_{dim} wytrenowany na N przypadkach uczących jest ograniczony przez sumę błędu na zbiorze uczącym $E_T(h)$ oraz wyrażenia zależnego od liczby próbek N i wymiaru VC_{dim}

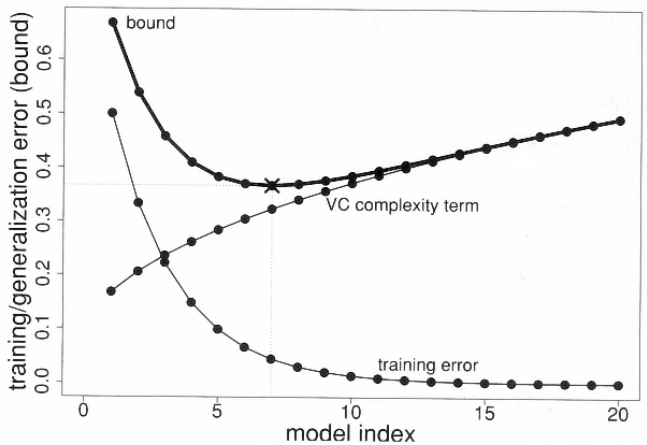
$$E_G(h) \leq E_T(h) + \sqrt{\frac{\mathrm{VC}_{dim}(\mathbb{H}) \cdot \left(1 + \log\left(\frac{2N}{\mathrm{VC}_{dim}(\mathbb{H})}\right)\right) - \log(\frac{\delta}{4})}{N}}$$

z prawdopodobieństwem $1 - \delta$ jeżeli $\mathrm{VC}_{dim}(\mathbb{H}) \ll N$.

V. Vapnik (2000) The nature of statistical learning theory

Minimalizacja ryzyka strukturalnego

Structural risk minimization (SRM) (Vapnik) zasada równoważenia złożoności modelu względem jego zdolności do dopasowywania się do danych



- **im bardziej złożony model tym więcej potrzeba danych uczących do uzyskania satysfakcjonującego błędu generalizacji**
- jeśli znamy VC_{dim} klasyfikatora to możemy oszacować ile próbek uczących jest potrzebnych aby uzyskać satysfakcjonujący błąd generalizacji

Rys: https://julien-vitay.net/lecturenotes-neurocomputing/2-linear/6-LearningTheory.html

Dobór złożoności sieci

Rule of thumb

$$E_G(\mathbf{w}) \approx \frac{\mathrm{VC}_{dim}}{N}$$

- prawdopodobieństwo przeuczenia rośnie, gdy $\mathrm{VC}_{dim} > N$
- złożoność VC_{dim} sieci neuronowej jest proporcjonalna do liczby wag N_w
  $\mathrm{VC}_{dim} \approx N_w$ dla MLP z binarnymi funkcjami aktywacji
  $\mathrm{VC}_{dim} \approx 2N_w$ dla MLP z sigmoidalnymi funkcjami aktywacji
- regularyzacja zmniejsza VC_{dim} modelu
- dla złożonych modeli wyznaczenie VC_{dim} jest trudne
- w praktyce: minimalna liczba próbek uczących powinna być co najmniej 10-krotnością wymiaru VC

$$N \geq 10 VC_{dim}$$

Regularyzacja L_2

Regularyzacja L_2 (weight decay, regularyzacja Tichonova) - dąży do zmniejszenia wartości wag

$$\hat{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{1}{2}\lambda \sum_{ij} w_{ij}^2$$

gdzie $\lambda > 0$ decyduje o sile regularyzacji (typowo $\lambda = 0.01$ lub 0.001)

Zmniejszane są wszystkie wagi w trakcie uczenia.
Korekta wag $\hat{w}_{ij}$ w stosunku do wersji nieregularyzowanej w_{ij}

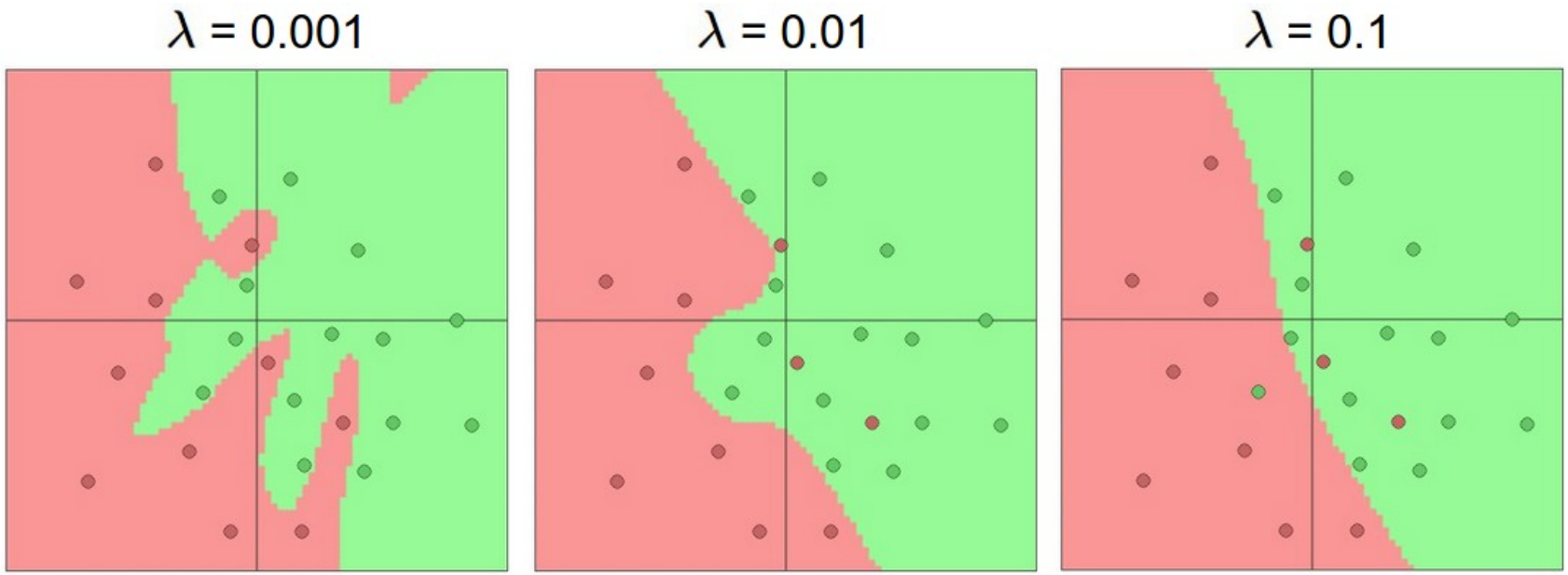
$$\hat{w}_{ij} = (1 - \lambda\eta) w_{ij}$$

$$\mathbf{w}(n+1) = \hat{\mathbf{w}}(n) - \eta \nabla E(\mathbf{w})$$

Możliwe modyfikacje:

- wagi, które zmalały poniżej pewnego progu mogą być usunięte
- usuwamy neurony dla których $\sum |w_i|$ jest bliskie zeru

Regularyzacja



Rys: http://cs231n.github.io/neural-networks-1/

Zmodyfikowana regularyzacja L_2

W przypadku regularyzacji L_2 wartości wszystkich wag maleją. Poniższa modyfikacja zachowuje wagi o dużych wartościach

$$\hat{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{1}{2}\lambda \sum_{ij} \frac{w_{ij}^2}{1 + w_{ij}^2}$$

Korekta wag $\hat{w}_{ij}$ w stosunku do wersji nieregularyzowanej w_{ij}

$$\hat{w}_{ij} = \left(1 - \lambda\eta \frac{1}{\left(1 + w_{ij}^2\right)^2}\right) w_{ij}$$

- dla małych wag $w_{ij} \ll 1$ równoważne z L_2
- dla dużych wag $w_{ij} \gg 1$ regularyzacja zanika

Zmodyfikowana regularyzacja L_2

Zmodyfikowany człon kary wykorzystywany do eliminacji niepotrzebnych neuronów w warstwie:

$$\hat{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{1}{2}\lambda \sum_{ij} \frac{w_{ij}^2}{1 + \sum_k w_{ik}^2}$$

Minimalizacja E powoduje:

- eliminacje połączeń (wagi zanikają)
- eliminację neuronów dla których $\sum_k |w_{ik}|$ jest bliskie zera

Regularyzowana zmiana wag zależna jest od wartości w_{ij}

$$\hat{w}_{ij} = \left(1 - \lambda\eta \frac{1 + 2\sum_{k \neq j} w_{ik}^2}{\left(1 + \sum_k w_{ik}^2\right)^2} \right) w_{ij}$$

- małe wartości wag w_{ik} prowadzące do i-tego neuronu zanikają
- duże wartości wag $\sum_k |w_{ik}|$ neuronu niwelują wpływ czynnika korekcyjnego

Regularyzacja L_1

$$\hat{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \sum_{ij} |w_{ij}|$$

$$\nabla \hat{E}(\mathbf{w}) = \nabla E(\mathbf{w}) + \lambda \operatorname{sgn}(\mathbf{w})$$

regularyzacja wpływa na gradient ze stałą wartością (zależy tylko od znaku w_{ij})

$$\hat{w}_{ij} = \begin{cases} w_{ij} - \lambda\eta & \text{dla} \quad w_{ij} > 0 \\ w_{ij} + \lambda\eta & \text{dla} \quad w_{ij} < 0 \end{cases}$$

Regularyzacja L_1 daje rzadsze rozwiązanie od L_2, tzn. dąży do rozwiązania o mniejszej liczbie niezerowych wag

Metody wrażliwościowe redukcji sieci

- Eliminacja wag na podstawie ich wpływu na wartość błędu
- Wagi o małej wartości $|w_i|$ są mniej istotne od dużych wag
- Wrażliwość połączenia wyznaczana dla nauczonej sieci o błędzie E

  $$S_{ij} = E - E(w_{ij} = 0)$$

  gdzie $E(w_{ij} = 0)$ to błąd sieci po usunięciu w_{ij}
- Połączenia o najmniejszej wrażliwości usuwa się po czym sieć jest douczana
- Usuwamy neuron dla którego wszystkie dochodzące (lub wychodzące) połączenia są wyzerowane

Optimal Brain Damage (LeCun)

Założenie: hesjan $\mathbf{H}$ jest diagonalnie dominujący, więc uwzględniamy tylko składową diagonalną.

Z rozwinięcia funkcji błędu E w szereg Tylora i przyjmując $\nabla E = 0$ (punkt zbieżności dla wytrenowanej sieci) otrzymujemy oszacowanie zmiany błędu:

$$\Delta E \approx \frac{1}{2}\Delta \mathbf{w}^T \mathbf{H} \Delta \mathbf{w} = \frac{1}{2}\sum_i H_{ii}\Delta w_i^2$$

Współczynnik istotności wagi

$$S_i = \frac{1}{2}\frac{\partial^2 E}{\partial w_i^2} w_i^2$$

Wagi o najmniejszym S_i mogą być usunięte z wytrenowanej sieci.

Optimal Brain Damage

Algorytm redukcji sieci OBD

1. Wstępne uczenie: sieć jest w pełni uczona na dostępnych danych, aż osiągnie lokalne minimum funkcji celu
2. Wyznacz diagonalne elementy hesjanu oraz wartości wrażliwości S_i dla każdej wagi w_i
3. Posortuj wagi zgodnie z rosnącymi wartościami S_i i obetnij te o najmniejszych wartościach (zazwyczaj 1-5% wag)
4. Wróć do punktu 1

Redukcja neuronów o małej aktywności

- Neurony o niewielkiej aktywności (których sygnał wyjściowy nie zmienia się dla całego zbioru treningowego) można usunąć bez szkody dla generalizacji
- duża aktywność neuronu świadczy o jego przydatności
- modyfikacja funkcji kosztu z czynnikiem kary za zbyt małą aktywność neuronów (Y. Chaurin)

$$\hat{E} = E + \lambda \sum_{i=1}^{k} \sum_{j=1}^{n} e(\Delta_{ij}^2)$$

gdzie Δ_{ij} oznacza zmianę sygnału wyjściowego i-tego neuronu dla j-tego wektora treningowego

- przykładowy czynnik korelacyjny

$$e = \frac{1}{1 + \Delta_{ij}^2}$$