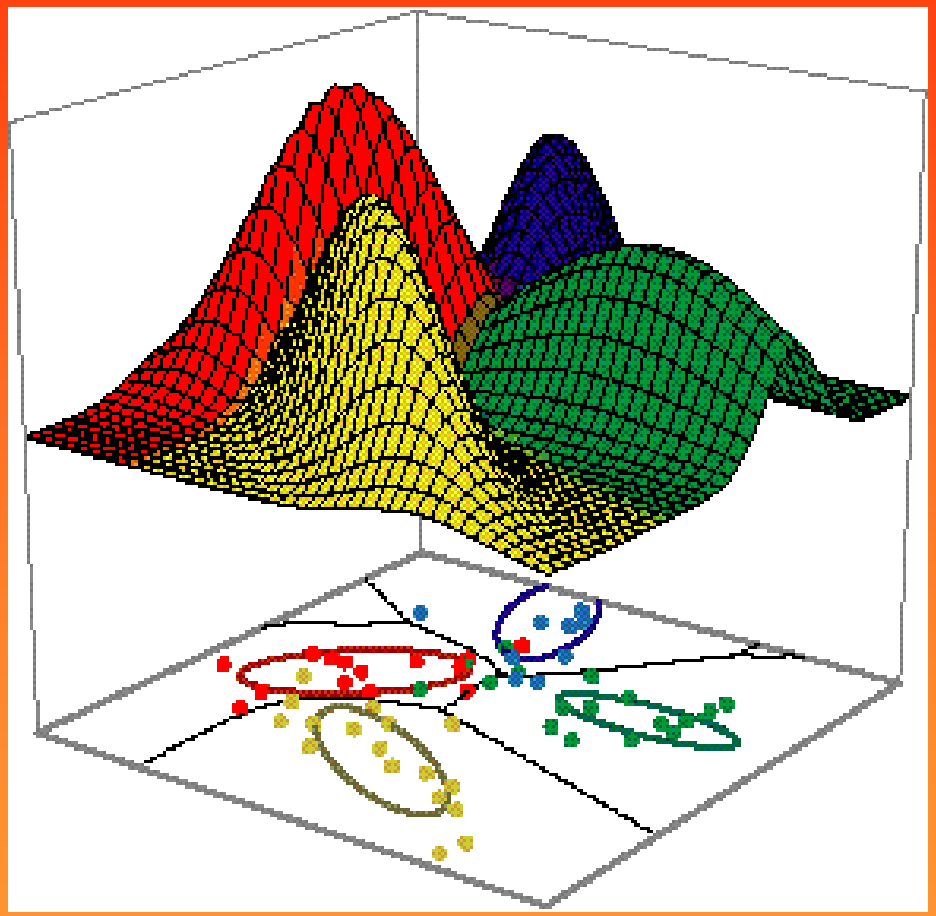


Pattern Classification

All materials in these slides were taken from
Pattern Classification (2nd ed) by R. O. Duda, P. E. Hart and D. G. Stork, John Wiley & Sons, 2000
with the permission of the authors and the publisher

Chapter 6: Wielowarstwowe sieci neuronowe (Sections 6.1-6.3)

- Wprowadzenie
- Propagacja sygnałów i klasyfikacja
- Algorytm wstecznej propagacji błędów

Wprowadzenie

- Cel: Klasyfikacja obiektów poprzez uczenie charakterystyk nieliniowych
  - W wielu zadaniach liniowe funkcje dyskryminujące dają zbyt duże wartości funkcji kryterialnej
  - Poprzednio, istotą rozwiązania był odpowiedni wybór zestawu funkcji nieliniowych
  - Teoretycznie, najlepiej byłoby zastosować kompletną postać funkcji dyskryminującej, np. wielomian dowolnego rzędu; ale wiązałoby się to z koniecznością estymacji zbyt wielkiej liczby parametrów na podstawie stosunkowo niewielkiej liczby przykładów z ciągu uczącego

- Nie jest znana metoda automatycznego doboru „prawdziwej” charakterystyki nieliniowej, jeżeli nie dysponujemy żadną dodatkową informacją o tej charakterystyce

- Sieci neuronowe „uczą się” postaci charakterystyki nieliniowej na podstawie danych uczących

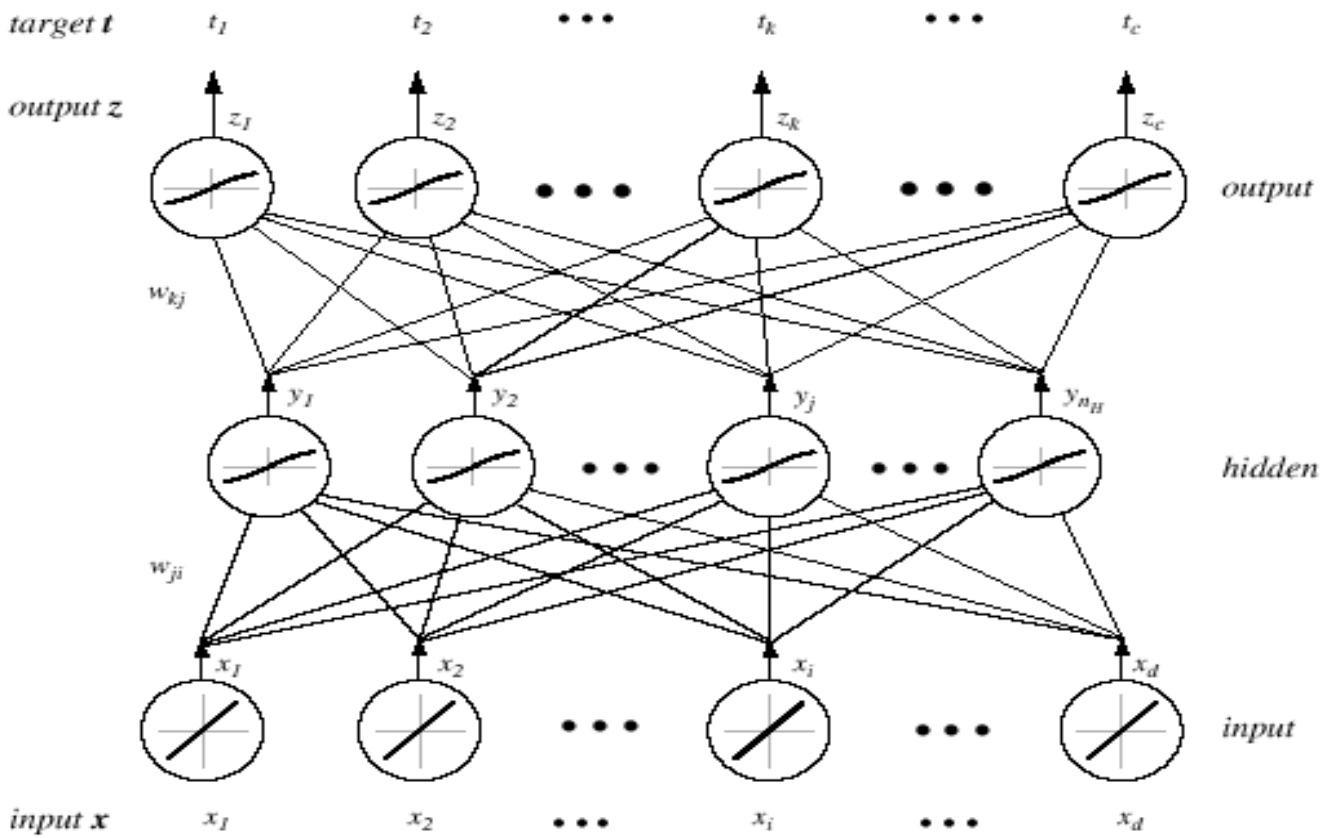
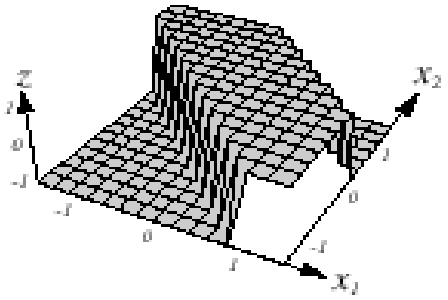
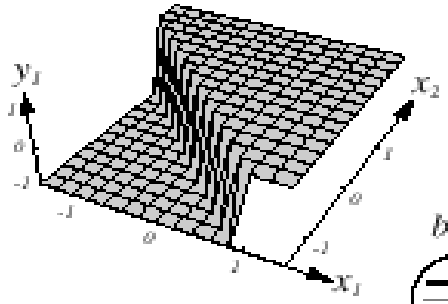


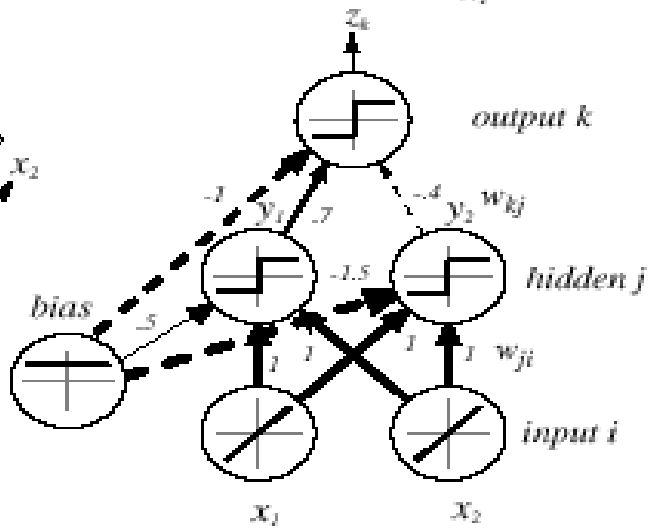
FIGURE 6.4. A d-n_H-c fully connected three-layer network and the notation we shall use. During feedforward operation, a d-dimensional input pattern $\mathbf{x}$ is presented to the input layer; each input unit then emits its corresponding component x_i. Each of the n_H hidden units computes its net activation, net_j, as the inner product of the input layer signals with weights w_{ji} at the hidden unit. The hidden unit emits $y_j = f(net_j)$, where $f(\cdot)$ is the nonlinear activation function, shown here as a sigmoid. Each of the c output units functions in the same manner as the hidden units do, computing net_k as the inner product of the hidden unit signals and weights at the output unit. The final signals emitted by the network, $z_k = f(net_k)$, are used as discriminant functions for classification. During network training, these output signals are compared with a teaching or target vector $\mathbf{t}$, and any difference is used in training the weights throughout the network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

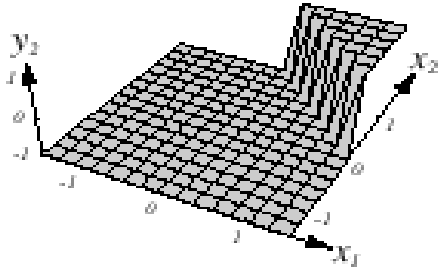
Operacje jednokierunkowe i klasyfikacja

- 3-warstwowa sieć neuronowa składa się z warstw: wejściowej, ukrytej i wyjściowej. Połączeniom pomiędzy kolejnymi warstwami odpowiadają parametry (nazywane wagami), których wartości można modyfikować


z
x2
x1


y1
x2
x1


zk
output k
wkj
hidden j
bias
wji
input i
x1
x2


y2
x2
x1

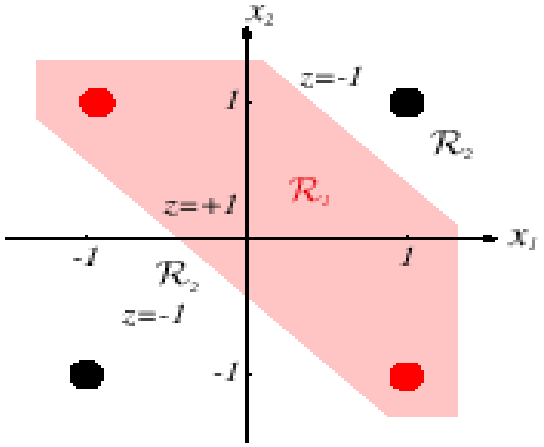

x2
z=-1
R2
z=+1
R1
x1
R2
z=-1

FIGURE 6.1. The two-bit parity or exclusive-OR problem can be solved by a three-layer network. At the bottom is the two-dimensional feature $x_1 x_2$-space, along with the four patterns to be classified. The three-layer network is shown in the middle. The input units are linear and merely distribute their feature values through multiplicative weights to the hidden units. The hidden and output units here are linear threshold units, each of which forms the linear sum of its inputs times their associated weight to yield *net*, and emits a +1 if this *net* is greater than or equal to 0, and −1 otherwise, as shown by the graphs. Positive or "excitatory" weights are denoted by solid lines, negative or "inhibitory" weights by dashed lines; each weight magnitude is indicated by the line's thickness, and is labeled. The single output unit sums the weighted signals from the hidden units and bias to form its *net*, and emits a +1 if its *net* is greater than or equal to 0 and emits a −1 otherwise. Within each unit we show a graph of its input-output or activation function—$f(net)$ versus *net*. This function is linear for the input units, a constant for the bias, and a step or sign function elsewhere. We say that this network has a 2-2-1 fully connected topology, describing the number of units (other than the bias) in successive layers. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- W celu ułatwienia zapisu, do każdego neuronu doprowadzane jest dodatkowe wejście o wartości równej 1

- Aktywacja neuronu (wyjście jego części liniowej):

$$net_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} \equiv \mathbf{w}_j^T \mathbf{x},$$

- gdzie *i* oznacza numer wejścia, *j* – numer neuronu w warstwie ukrytej; w_{ji} to waga połączenia między *j*-tym neuronem warstwy ukrytej a *i*-tym wejściem.

- Każdy neuron warstwy ukrytej zwraca sygnał wyjściowy y_j będący nieliniową funkcją jego aktywacji, tj: $y_j = f(net_j)$

Na poprzednim rysunku zilustrowano funkcję progową

$$f(net) = \mathrm{sgn}(net) \equiv \begin{cases} 1 \text{ if net} \geq 0 \\ -1 \text{ if net} < 0 \end{cases}$$

- Funkcja *f(.)* nazywana jest funkcją aktywacji lub nieliniowością neuronu. Istnieją różne – bardziej ogólne – postacie funkcje aktywacji

- Każdy neuron warstwy wyjściowej wyznacza swoją aktywację, przy czym na jego wejścia podawane są wyjścia neuronów warstwy ukrytej:

$$net_k = \sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} = \mathbf{w}_k^T \mathbf{y},$$

gdzie *k* oznacza numer neuronu warstwy wyjściowej a n_H jest numerem neurony warstwy ukrytej

- Neurony warstwy wyjściowej realizują nieliniową funkcję ich aktywacji, zwracając na wyjściu sygnał:

$$z_k = f(net_k)$$

- W przypadku *c* wyjść (klas), można rozpatrywać sieć jako układ *c* funkcji dyskryminujących $z_k = g_k(x)$ i klasyfikujący obraz x na podstawie tej funkcji, $g_k(x)$ $\forall k = 1, \ldots, c$, która przyjmuje największą wartość dla danego x

- 3-warstwowa sieć neuronowa z wagami podanymi na rys. 6.1 rozwiązuje problem XOR

- Neuron warstwy wejściowej y_1 określa granicę:

$x_1 + x_2 + 0.5 = 0$

$\geq 0 \Rightarrow y_1 = +1$

$< 0 \Rightarrow y_1 = -1$

- Neuron warstwy ukrytej y_2 określa granicę :

$x_1 + x_2 - 1.5 = 0$

$\geq 0 \Rightarrow y_2 = +1$

$< 0 \Rightarrow y_2 = -1$

- Neuron warstwy wyjściowej zwraca sygnał

  $z_1 = +1 \Leftrightarrow y_1 = +1$ *and* $y_2 = +1$

  $z_k = y_1$ i nie $y_2 = (x_1$ lub $x_2)$ i nie $(x_1$ i $x_2) = x_1$ *XOR* x_2

- Operacje jednokierunkowe – przypadek c wyjść

$$g_k(\mathbf{x}) \equiv z_k = f\left(\sum_{j=1}^{n_H} w_{kj} f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right) \qquad (1)$$

$$(k = 1,\ldots,c)$$

- Warstwa ukryta umożliwia użycie bardziej skomplikowanych postaci nieliniowości, rozszerzając możliwości klasyfikatora

- Można tak wybrać funkcję aktywacji, aby była ona ciągła i różniczkowalna

- Można dobrać różne funkcje aktywacji w warstwie ukrytej i wyjściowej, a nawet przyjąć, że każdy neuron ma indywidualnie dobraną postać funkcji aktywacji

- Dalej przyjmiemy, że wszystkie neurony sieci mają identyczną funkcję aktywacji

- Zdolność objaśniająca sieci wielowarstwowych

Pytanie: Czy 3-warstwowa sieć neuronowa opisana równaniem (1) może zrealizować dowolny zbiór reguł decyzyjnych?

Odpowiedź: Tak(A. Kołmogorow)

"Każda ciągła funkcja może być z dowolną dokładnością aproksymowana przez 3-wartwową sieć neuronową, mającą odpowiednią liczbę neuronów n_H w warstwie ukrytej, odpowiednie postacie funkcji aktywacji i prawidłowo dobrane wartości wag."

$$g(\mathbf{x}) = \sum_{j=1}^{2n+1} \delta_j \left(\Sigma \beta_{ij} (x_i) \right) \quad \forall \mathbf{x} \in I^n (I = [0,1]; n \geq 2)$$

- Na wejście każdego z *2n+1* neuronów warstwy ukrytej δ_j podawana jest suma *d* nieliniowych funkcji, jedna na każdą cechę x_i

- Każdy neuron ukryty zwraca wartość nieliniowej funkcji δ_j dla podanych wejść

- Neuron wyjściowy zwraca sumę wkładów wnoszonych przez neurony warstwy ukrytej

  Niestety twierdzenie Kołmogorowa nie daje żadnych wskazówek jak dobrać postać nieliniowej funkcji aktywacji do danych pomiarowych;

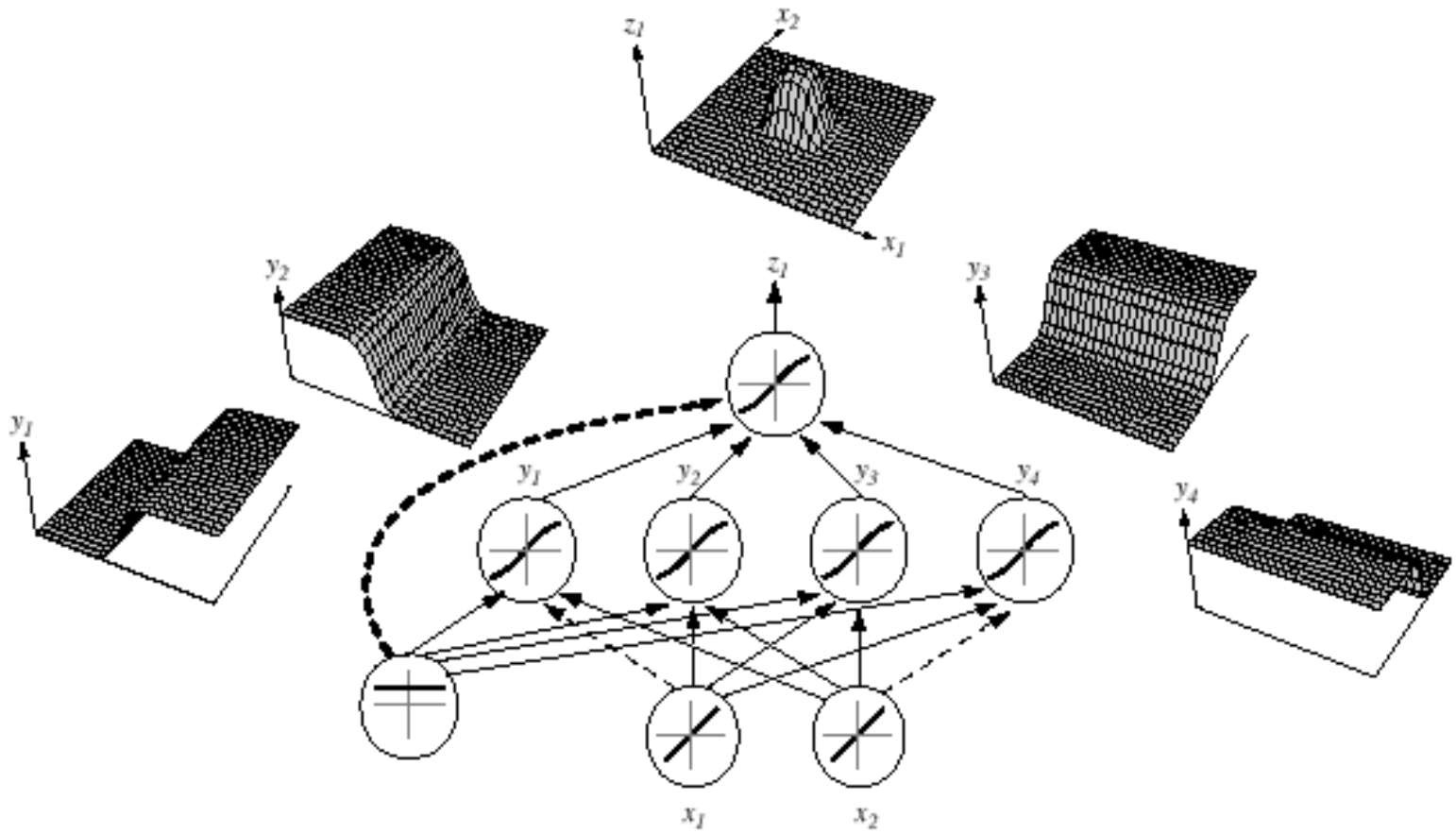


FIGURE 6.2. A 2-4-1 network (with bias) along with the response functions at different units; each hidden output unit has sigmoidal activation function $f(\cdot)$. In the case shown, the hidden unit outputs are paired in opposition thereby producing a "bump" at the output unit. Given a sufficiently large number of hidden units, any continuous function from input to output can be approximated arbitrarily well by such a network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Algorytm wstecznej propagacji błędów (BP – *BackPropagation*)

- Każda funkcja wiążąca wejście z wyjściem może być zrealizowana w postaci 3-warstwowej sieci neuronowej

- To twierdzenia ma dużą wartość, ale tylko dla teorii, ponieważ nie mówi nic o postaciach funkcji aktywacji, liczbach neuronów w poszczególnych warstwach o ani wartościach wag sieci.

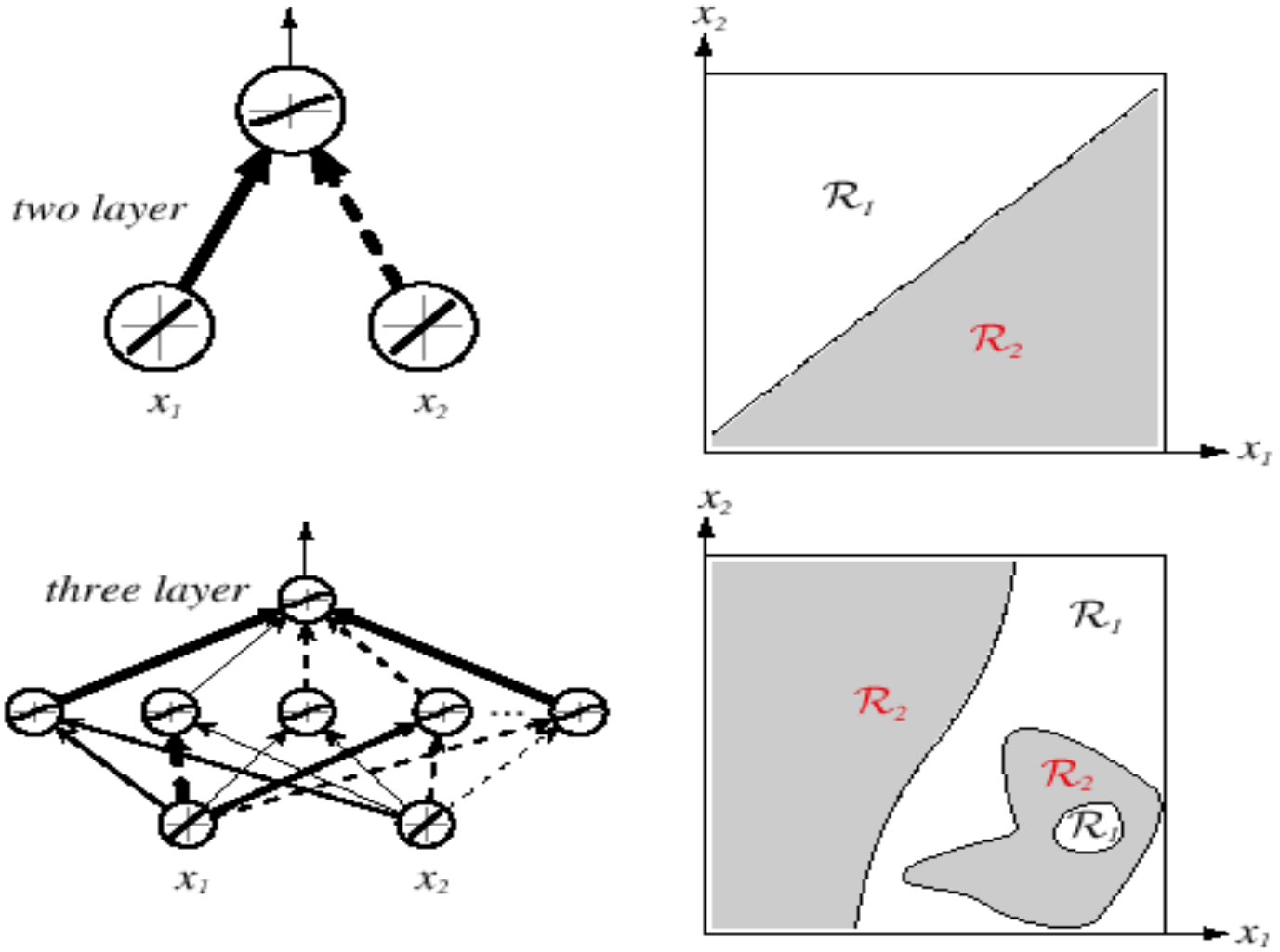


FIGURE 6.3. Whereas a two-layer network classifier can only implement a linear decision boundary, given an adequate number of hidden units, three-, four- and higher-layer networks can implement arbitrary decision boundaries. The decision regions need not be convex or simply connected. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- Dalej skupimy się na zadaniu doboru wartości wag sieci na podstawie ciągu uczącego

- W sieci 3-warstowej łatwo jest ustalić związek między błędem (a zarazem wyjściem) i wagami warstwy ukrytej

- Algorytm BP pozwala wyznaczyć błędy popełniane przez neurony warstwy ukrytej, co ułatwia określenie reguły uczenia wag połączeń między wejściami sieci i neuronami warstwy ukrytej (*the credit assignment problem*)

- Wyróżniamy dwa tryby działania sieci:
  - Propagacja sygnałów

    Propagacja sygnałów polega na podawaniu na wejście sieci obrazu i przetwarzaniu sygnałów przez kolejne warstwy sieci aż do jej wyjścia (bez cykli)
  - Uczenie

    Uczenie z nauczycielem polega na pobudzaniu wejścia sieci wzorcem z ciągu uczącego a następnie takiej modyfikacji wartości wag, aby zmniejszyła się różnica między wyjściem sieci a prawidłowym wyjściem, odpowiadającym podanemu wzorcowi

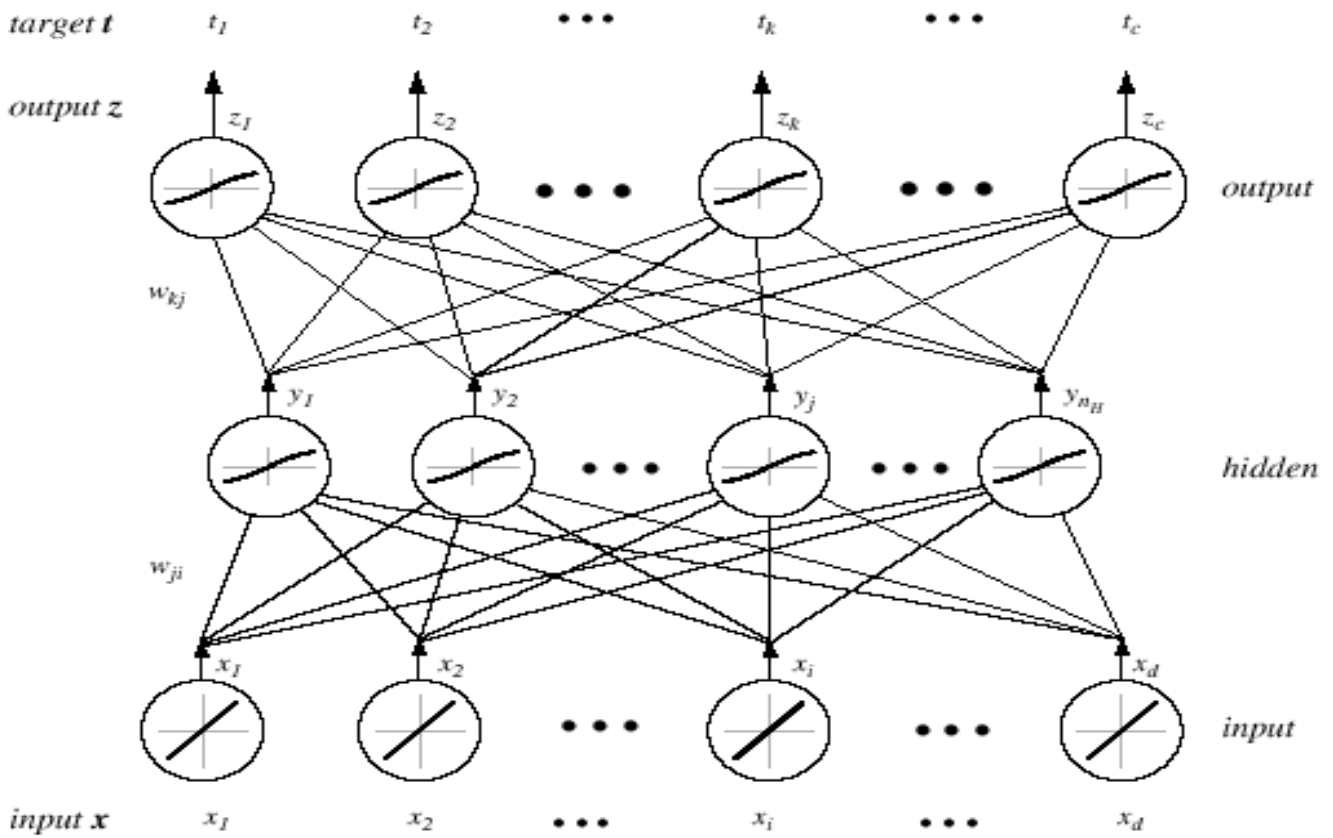


FIGURE 6.4. A d-n_H-c fully connected three-layer network and the notation we shall use. During feedforward operation, a d-dimensional input pattern $\mathbf{x}$ is presented to the input layer; each input unit then emits its corresponding component x_i. Each of the n_H hidden units computes its net activation, net_j, as the inner product of the input layer signals with weights w_{ji} at the hidden unit. The hidden unit emits $y_j = f(net_j)$, where $f(\cdot)$ is the nonlinear activation function, shown here as a sigmoid. Each of the c output units functions in the same manner as the hidden units do, computing net_k as the inner product of the hidden unit signals and weights at the output unit. The final signals emitted by the network, $z_k = f(net_k)$, are used as discriminant functions for classification. During network training, these output signals are compared with a teaching or target vector $\mathbf{t}$, and any difference is used in training the weights throughout the network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- Uczenie sieci neuronowej

  - Niech t_k będzie *k*-tym prawidłowym wyjściem sieci a z_k *k*-tym wyjściem obliczonym przez sieć, $k = 1, \ldots, c$, w – to wektor wszystkich wag sieci

  - Błąd uczenia:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}\|\mathbf{t} - \mathbf{z}\|^2$$

  - Algorytm BP jest oparty na metodzie gradientu prostego (numeryczna metoda optymalizacji)
    - Początkowe wartości wag są dobierane losowo a modyfikuje się je zgodnie z kierunkiem, w którym błąd maleje:

$$\Delta\mathbf{w} = -\eta\frac{\partial J}{\partial \mathbf{w}}$$

gdzie η to **współczynnik uczenia** wpływający na wielkość modyfikacji wartości wag:

$$w(m+1) = w(m) + \Delta w(m)$$

gdzie m jest m-tym obrazem prezentowanym sieci

- Błąd popełniany przez neurony warstwy wyjściowej:

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}}$$

gdzie $\delta_k = -\frac{\partial J}{\partial net_k}$ jest wrażliwością k-tego neuronu

i opisuje wpływ aktywacji na zmiany błędu:

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} = (t_k - z_k) f'(net_k)$$

Ponieważ $net_k = w_k^T y$, to:

$$\frac{\partial net_k}{\partial w_{kj}} = y_j$$

Wniosek: reguła uczenia (reguła aktualizacji wag) dla wag połączeń między warstwą wyjściową i ukrytą jest następująca:

$$\Delta w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k)\, f'(net_k) y_j$$

- Błąd popełniany przez neurony warstwy ukrytej

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}}$$

Jednakże,

$$\frac{\partial J}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2\right] = -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j}$$

$$= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial net_k}.\frac{\partial net_k}{\partial y_j} = -\sum_{k=1}^{c}(t_k - z_k) f'(net_k) w_{kj}$$

Analogicznie jak poprzednio, definiujemy wrażliwość neuronu warstwy ukrytej:

$$\delta_j \equiv f'(net_j)\sum_{k=1}^{c} w_{kj}\delta_k$$

co oznacza, że:"wrażliwość neuronu warstwy ukrytej jest sumą wrażliwości powiązanych z nim neuronów warstwy wyjściowej, ważoną przez wagi w_{kj} tych połączeń i przeskalowaną przez $f'(net_j)$"

Wniosek: reguła uczenia dla wag połączeń między warstwą ukrytą i wejściową jest następująca:

$$\Delta w_{ji} = \eta x_i \delta_j = \eta \underbrace{\left[\Sigma w_{kj}\delta_k\right] f'(net_j)}_{\delta_j} x_i$$

- Algorytm BP:

```
Begin  initialize    n_H; w, criterion θ, η, m ← 0
       do m ← m + 1
            x^m ← randomly chosen pattern
            w_ji ←  w_ji + ηδ_j x_i; w_kj ← w_kj + ηδ_k y_j
       until ||∇J(w)|| < θ
        return w
End
```

- Kryterium stopu
  - Algorytm BP kończy działanie, gdy zmiany wartości funkcji kryterialnej *J(*w*)* są mniejsze od przyjętego progu θ
  - Istnieją inne kryteria stopu, przydatne w różnych sytuacjach
  - Dotychczas rozważaliśmy błąd klasyfikacji dla pojedynczego obrazu. Jak określić błąd dla całego ciągu uczącego?
  - Całkowity błąd dla ciągu uczącego jest sumą błędów popełnianych dla *n* pojedynczych obrazów

$$J = \sum_{p=1}^{n} J_p \qquad (1)$$

- Kryterium stopu (ciąg dalszy)
  - Modyfikując wagi, zmniejszamy błąd dla bieżącego wzorca, ryzykując zwiększenie błędów dla pozostałych wzorców
  - Jednak po wielu iteracjach całkowity błąd (1) będzie się zmniejszać z bardzo dużym prawdopodobieństwem

- Krzywe uczenia
  - Na początku procesu uczenia błąd dla ciągu uczącego jest duży; w trakcie kolejnych iteracji ten błąd staje się mniejszy
  - Przebieg błędu (krzywa uczenia) dla wybranego wzorca zależy od liczby wzorców w ciągu uczącym i zdolności objaśniającej (np. liczba wag) sieci
  - Średni błąd dla niezależnego ciągu testującego jest zawsze większy od błędu dla ciągu uczącego i może się zarówno zmniejszać jak i zwiększać w trakcie procesu uczenia
  - Ciąg walidacyjny pomaga w podjęciu decyzji o zatrzymaniu procesu uczenia; chcemy uniknąć nadmiernego dopasowania sieci do danych i zmniejszenia zdolności klasyfikatora do generalizowania

    "zatrzymujemy proces uczenia po osiągnięciu minimum błędu dla ciągu walidacyjnego"

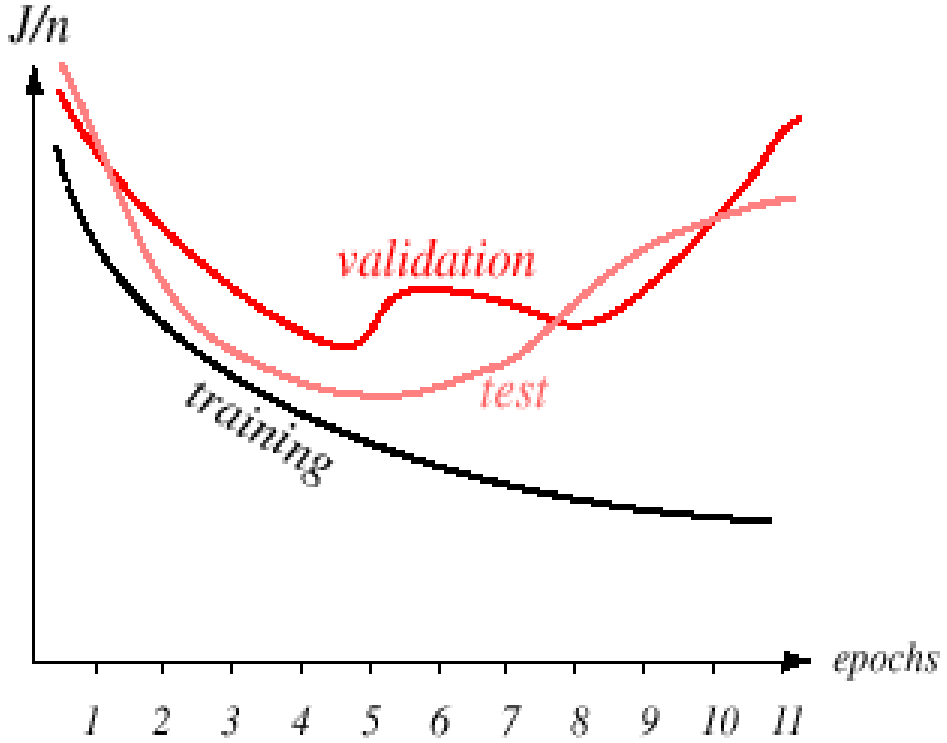


FIGURE 6.6. A learning curve shows the criterion function as a function of the amount of training, typically indicated by the number of epochs or presentations of the full training set. We plot the average error per pattern, that is, $1/n \sum_{p=1}^{n} J_p$. The validation error and the test or generalization error per pattern are virtually always higher than the training error. In some protocols, training is stopped at the first minimum of the validation set. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.