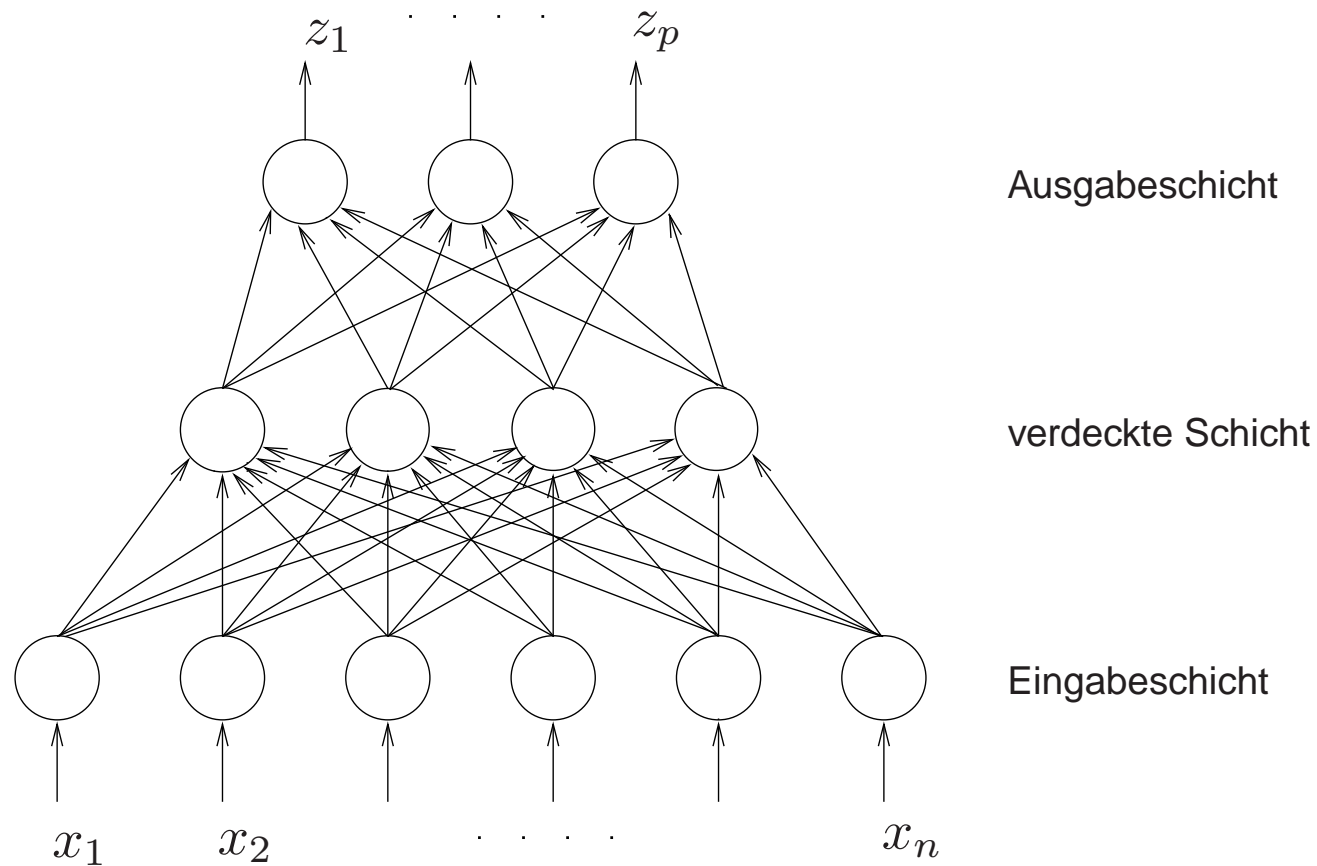
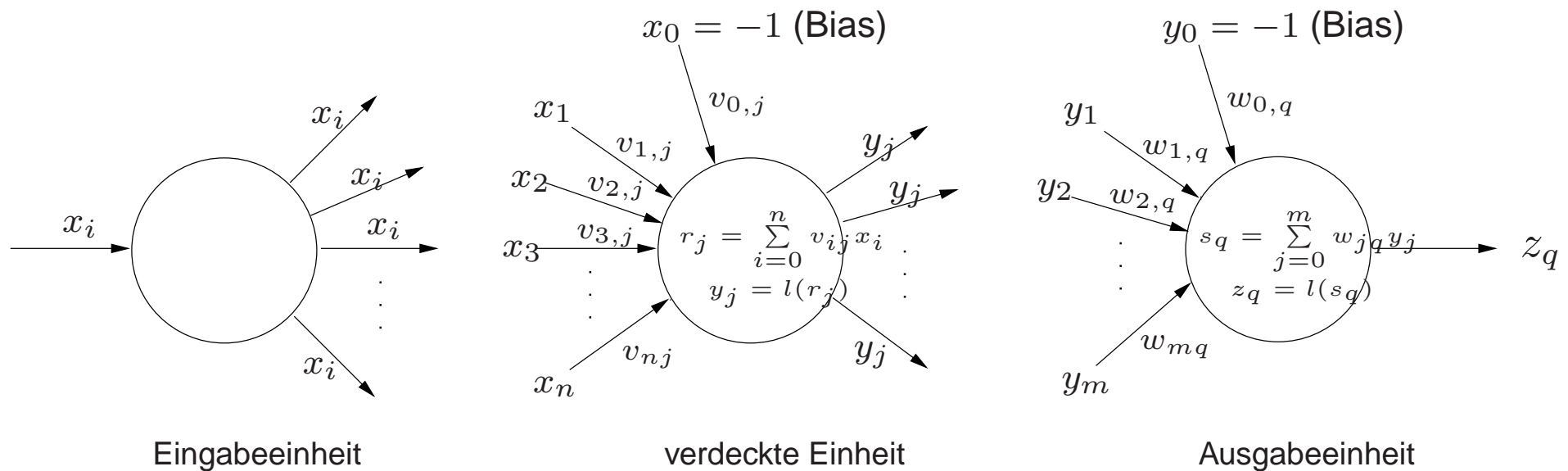


2.2 Vorwärtsgerichtete neuronale Netze



Vollständig verbundenes feed-forward Netz mit einer verdeckten Schicht



Die Einheiten der verschiedenen Schichten.

Ankommenden Werte bei der verdeckten Einheit und beim Ausgabeneuron werden mit l transformiert.

Der Wert des verdeckten Neurons bleibt für den Beobachter verborgen.

Jedes Neuron i der Eingabeschicht leitet ein bestimmtes Merkmal eines Objekts als binären (0/1 oder ± 1) oder reellen Wert x_i an jede Einheit der verdeckten Schicht mit unterschiedlichen Gewichten v_{ij} weiter.

Bias-Neuronen Nr. 0 haben Gewicht 1.

Jedes Neuron $j > 0$ der verdeckten Schicht erhält als Eingabe die gewichtete Summe der Ausgabe der vorangehenden Schicht

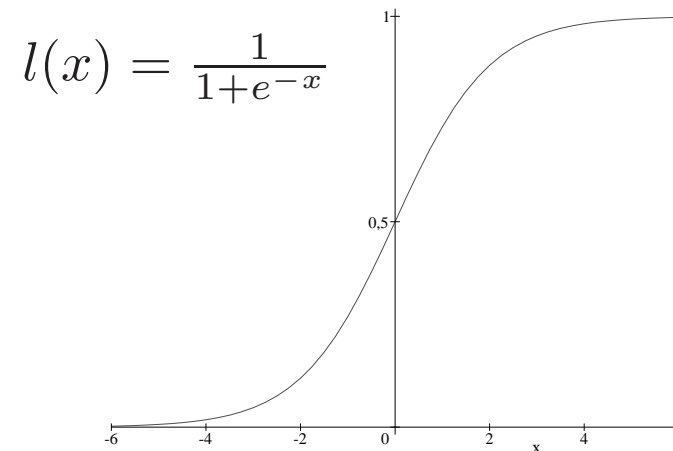
$$r_j = \sum_{i=0}^n v_{ij} x_i$$

und berechnet seine Ausgabe

$$y_j = a(r_j),$$

mit der *Aktivierungsfunktion* $a(\cdot)$,
z.B. die logistische Funktion $l(\cdot)$, d.h.

$$y_j = l(r_j) = \frac{1}{1 + e^{-r_j}}$$



w_{jq} : Gewicht an der Kante zwischen verdecktem Neuron j und Ausgabeneuron q .

Ausgabeneuron q erhält als Eingabe die gewichtete Summe

$$s_q = \sum_{j=0}^m w_{jq} y_j$$

und berechnet seine Ausgabe z_q mit

$$z_q = l(s_q).$$

Jedes Ausgabeneuron repräsentiert dann eine Klasse.

Ein Muster, dessen Daten in das Netz eingegangen sind, wird der Klasse zuzuordnen, deren Ausgabeneuron den höchsten Wert liefert.

Zu den Eingaben x_i erhalten wir im zweischichtigen Netz mit

$$z_q = l\left(\sum_j w_{jq} l\left(\sum_i v_{ij} x_i\right)\right) =: \phi_{q,v,w}(x)$$

die zur Klasse q gehörige Ausgabefunktion zu festen Gewichten v und w .

2.3 Das Lernverfahren Backpropagation

Daten seien von der Form $O = (X, T) \in \mathcal{X} \times \mathcal{T}$ mit Merkmalraum \mathcal{X} und

$$\mathcal{T} = \left\{ (t_0, \dots, t_{p-1}) \mid t_i \in \{0, 1\}, \sum t_i = 1 \right\}.$$

$o = (x, t)$ mit Muster x gehört der Klasse q an, wenn $t_q = 1$ gilt.

$\phi : \mathcal{X} \rightarrow [0, 1]^p$ sei die Ausgabefunktion des Netzes

Die Entscheidungsregel:

$$\hat{T}(x) = (0, \dots, 0, t_q = 1, 0, \dots, 0) \quad \Leftrightarrow \quad \phi_q \geq \phi_r \quad \forall r \neq q,$$

Definiere ϕ durch Minimieren der Fehlerfunktion

$$F(v, w) = \mathbf{E}\{\|\phi(X) - T\|^2\} = \sum_{q=0}^{p-1} \mathbf{E}\{|\phi_q(X) - T_q|^2\}$$

Minimiere F durch Gradientenabstieg: Verändere v_{ij} und w_{ij} im Schritt k um

$$\Delta_k v_{ij} = -\eta \frac{\partial}{\partial v_{ij}} F_k(v, w) \quad \text{und} \quad \Delta_k w_{jq} = -\eta \frac{\partial}{\partial w_{jq}} F_k(v, w)$$

Dabei ist

$$F_k(v, w) = \frac{1}{2} \sum_{q=0}^{p-1} (z_q^{(k)} - t_q^{(k)})^2$$

der beim k -ten Objekt beobachtete Fehler.

Die Schrittgröße η wird im Laufe des Verfahrens verkleinert.

$$\frac{\partial F_k}{\partial w_{jq}} = ?$$

$$F_k(v, w) = \frac{1}{2} \sum_{q=0}^{p-1} \left(z_q^{(k)} - t_q^{(k)} \right)^2$$

$$z_q^{(k)} = l(s_q), \quad s_q = \sum_{j=0}^m w_{jq} y_j$$

$$\frac{\partial F_k}{\partial w_{jq}} = \frac{\partial F_k}{\partial s_q^{(k)}} \frac{\partial s_q^{(k)}}{\partial w_{jq}} = \frac{\partial F_k}{\partial z_q^{(k)}} \frac{\partial z_q^{(k)}}{\partial s_q} \frac{\partial s_q}{\partial w_{jq}}$$

$$\frac{\partial F_k}{\partial z_q^{(k)}} = z_q^{(k)} - t_q^{(k)}, \quad \frac{\partial s_q}{\partial w_{jq}} = y_j$$

$$\frac{\partial z_q^{(k)}}{\partial s_q} = ?, \quad z_q^{(k)} = l(s_q) = \frac{1}{1 + e^{-s_q}}$$

$$l'(x) = \frac{-1}{(1 + e^{-x})^2} \cdot e^{-x} \cdot (-1) = \frac{1}{1 + e^{-x}} \cdot \frac{e^{-x}}{1 + e^{-x}} = l(x) \cdot (1 - l(x))$$

Also:

$$\frac{\partial z_q^{(k)}}{\partial s_q} = z_q^{(k)} \cdot (1 - z_q^{(k)})$$

$$\alpha_q^{(k)} = \frac{\partial F_k}{\partial s_q^{(k)}} = \frac{\partial F_k}{\partial z_q^{(k)}} \frac{\partial z_q^{(k)}}{\partial s_q^{(k)}} = (z_q^{(k)} - t_q^{(k)}) z_q^{(k)} (1 - z_q^{(k)}).$$

heißt *Fehlersignal* am q-ten Ausgabeneuron im k -ten Lernschritt.

Wir erhalten damit

$$\Delta_k w_{jq} = -\eta \frac{\partial F_k}{\partial w_{jq}} = -\eta \frac{\partial F_k}{\partial s_q^{(k)}} \cdot \frac{\partial s_q^{(k)}}{\partial w_{jq}} = -\eta \alpha_q^{(k)} y_j^{(k)}$$

Nun zu

$$\frac{\partial F_k}{\partial v_{ij}} = \frac{\partial F_k}{\partial r_j^{(k)}} \frac{\partial r_j^{(k)}}{\partial v_{ij}}.$$

Wir erinnern uns:

$$F_k(v, w) = \frac{1}{2} \sum_{q=0}^{p-1} \left(z_q^{(k)} - t_q^{(k)} \right)^2$$

$$z_q = l \left(\sum_j w_{jq} y_j \right), \quad y_i = l(r_j), \quad r_j = \sum_i v_{ij} x_i$$

Also

$$\frac{\partial r_j^{(k)}}{\partial v_{ij}} = x_i^{(k)}$$

In das Fehlersignal am j -ten Verdeckten Neuron ist

$$\begin{aligned} \delta_j^{(k)} &:= \frac{\partial F_k}{\partial r_j^{(k)}} = \frac{\partial y_j^{(k)}}{\partial r_j^{(k)}} \frac{\partial F_k}{\partial y_j^{(k)}} = y_j^{(k)} (1 - y_j^{(k)}) \sum_q w_{jq} z_q^{(k)} (1 - z_q^{(k)}) (z_q^{(k)} - t_q^{(k)}) \\ &= y_j^{(k)} (1 - y_j^{(k)}) \sum_q w_{jq} \alpha_q^{(k)}. \end{aligned}$$

Wir erhalten also:

$$\Delta_k v_{ij} = -\eta x_i^{(k)} \delta_j^{(k)} \quad \text{für } j > 0,$$

mit

$$\delta_j^{(k)} = y_j^{(k)} (1 - y_j^{(k)}) \sum_q w_{jq} \alpha_q^{(k)}.$$

Auch bei mehr als einer verdeckten Schicht lassen sich die Fehlersignale aus denen der jeweils höheren Schicht berechnen. Daher der Name “Backpropagation”.

Eine Backpropagation-Iteration für ein N -schichtiges Netz:

1. Dem Netz wird ein Muster $x^{(k)}$ eines zufällig ausgewählten Trainingsobjekts $o^{(k)} = (x^{(k)}, t^{(k)})$ präsentiert: $v_i^0 = x_i^{(k)}$
2. Es verarbeitet die eingegangene Information *vorwärts* von Schicht zu Schicht weiter gemäß $v_j^n = l\left(\sum_i w_{ij}^n v_i^{n-1}\right)$,
3. Es erfährt den wahren Typ $t^{(k)}$ und berechnet die Fehlersignale $\delta^N = \alpha$ der Ausgabeneuronen: $\delta_j^N = v_j^N (1 - v_j^N)(v_j^N - t_j^{(k)})$
4. Es berechnet die Fehlersignale *rückwärts* von Schicht zu Schicht gemäß $\delta_i^{n-1} = v_i^{n-1} (1 - v_i^{n-1}) \sum_j w_{ij}^n \delta_j^n$ für $n = N, N - 1, \dots, 2$.
5. Es ändert seine Gewichte um den Zuwachs $\Delta w_{ij}^n = -\eta \delta_j^n v_i^{n-1}$ für $n = 1, \dots, N$.

2.4 PAC-Lernen

PAC = Probably Approximately Correct

Ziel: Lerne aus Beispielen, so dass mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ der Fehler beim Klassifizieren höchstens ϵ ist.

Ist D eine W'keitsverteilung auf Ω und $C \in \mathcal{C}$ das zu lernende Konzept, so können wir den **Fehler** der Hypothese $H \in \mathcal{H}$ durch $F_D(C, H) := D((C \cup H) \setminus (C \cap H))$ definieren.

Fairness-Bedingung: Bewerte den Fehler mit derselben Verteilung D , gemäß der die Lernbeispiele erzeugt wurden.

PAC-Algorithmus für eine Konzeptklasse \mathcal{C}

Ablauf:

- Eingabe: Vertrauensparameter δ , Fehlerparameter ϵ , Länge n der Beispiele
- Bestimme Anzahl $s = s(\delta, \epsilon, n)$ der anzufordernden Beispiele.
- Lies s Beispiele an
- Gib ein $H \in \mathcal{H}$ aus.

Kriterium: Für alle $\epsilon, \delta \in (0, 1]$, $C \in \mathcal{C}$ und alle Verteilungen D auf Ω gilt:

$$W_{sD}(F_D(C, H) \leq \epsilon) \geq 1 - \delta$$

($W_{sD}(\cdot)$:= W'keit unter der Voraussetzung, dass die Lerndaten aus D kommen)

Falls $\mathcal{H} = \mathcal{C}$ und es leicht ist, für jede mit $C \in \mathcal{C}$ verträgliche Folge von Beispielen $\{(x_1, b_1), \dots, (x_n, b_n)\} \subset \Omega \times \{0, 1\}$ (d.h. $b_i = 1 \Leftrightarrow x_i \in C$) eine konsistente Hypothese $H \in \mathcal{H}$ zu finden (d.h. $b_i = 1 \Leftrightarrow x_i \in H$), dann

- setze $s := \lceil (\ln(|\mathcal{C}|) - \ln(\delta)) / \varepsilon \rceil$
- lies s Beispiele ein
- Gib dazu konsistente Hypothese H aus

Satz 3 *Damit erhalten wir einen PAC-Algorithmus.*

Beweis:

$$\begin{aligned} \mathbb{W}_{s_D}(F_D(C, H) > \varepsilon) &\leq \mathbb{W}_{s_D}(\exists \text{ konsistentes } H \in \mathcal{C} : F_D(C, H) > \varepsilon) \\ &\leq \sum_{H \in \mathcal{C} : F_D(C, H) > \varepsilon} \mathbb{W}_{s_D}(H \text{ konsitent mit allen } (x_i, b_i)) \\ &\leq |\mathcal{C}|(1 - \varepsilon)^s \end{aligned}$$

Wähle also s so groß, dass $|\mathcal{C}|(1 - \varepsilon)^s \leq \delta$ gilt.

Logarithmieren ergibt $\ln |\mathcal{C}| + s \ln(1 - \varepsilon) \leq \ln \delta$ und wegen $\ln(1 - \varepsilon) \leq -\varepsilon$ genügt $s \geq (\ln |\mathcal{C}| - \ln \delta) / \varepsilon$. □