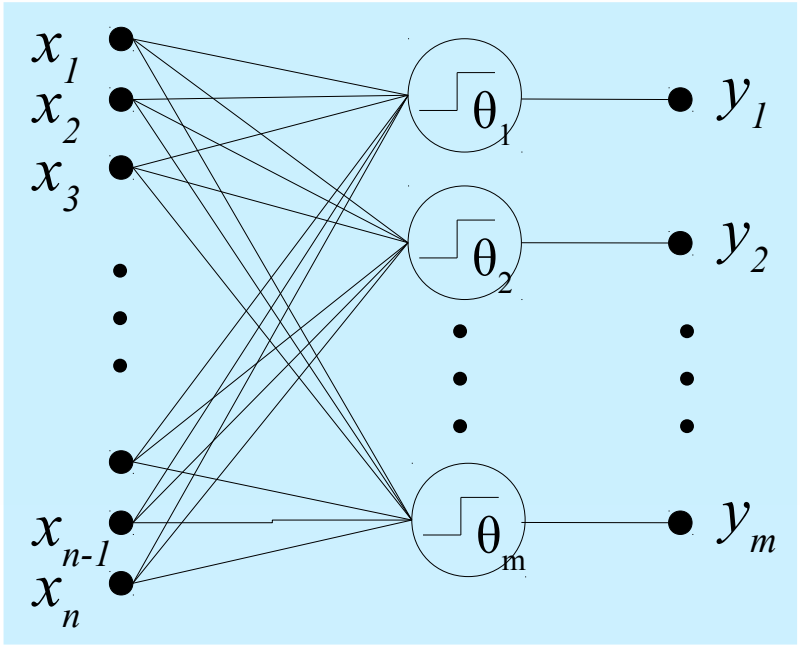
Fachhochschule
Münster University of
Applied Sciences

Adaptive Systeme

Neuronale Netze: der Backward Propagation Algorithmus

Prof. Dr. rer. nat. Nikolaus Wulff

Neuronen Schicht



$$y_j = \sigma\left(\sum_{k=1}^{n} w_{jk} x_k - \theta_j\right)$$

$$y_j = \sigma\left(\sum_{k=0}^{n} w_{jk} x_k\right)$$

$$\vec{y} = \sigma\left(W \vec{x}\right)$$

- Werden *m* Neuronen zu einer Schicht mit *n* gemeinsamen Eingängen zusammengefasst, so werden den *n* Eingangssignalen *m* Ausgangssignale zugeordnet.
- Es entsteht eine Abbildung $f: \{0,1\}^n \rightarrow \{0,1\}^m$.

Lernregel für ein Layer

- Mit einer stetig differenzierbaren Transferfunktion gilt für ein beliebiges Musterpaar (x,d):

$$\frac{\partial E}{\partial w_{jk}} = \left(\sigma\left(\sum_i w_{ji} x_i\right) - d_j\right) \sigma'\left(\sum_i w_{ji} x_i\right) x_k$$

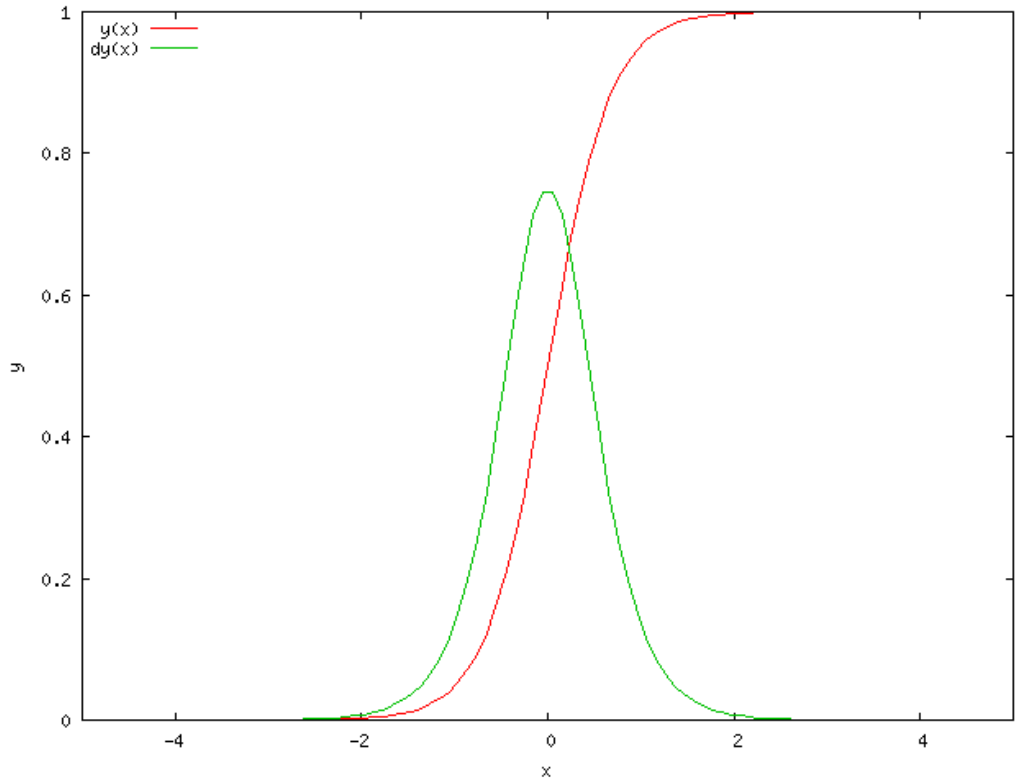
- und die Anwendung des Gradientenverfahrens liefert die Lernregel:

$$\Delta w_{jk} = \mu \left(d_j - y_j\right) \sigma'(\tilde{y}_j)\, x_k = \mu\, e_j\, \sigma'(\tilde{y}_j)\, x_k$$

- Die Bedingung $y_j \in \{0,1\}$ ist jetzt nicht mehr streng gegeben, sondern die Bildmenge ist das Intervall $[0,1]$

Ableitung der Transferfunktion

Fachhochschule
Münster University of
Applied Sciences



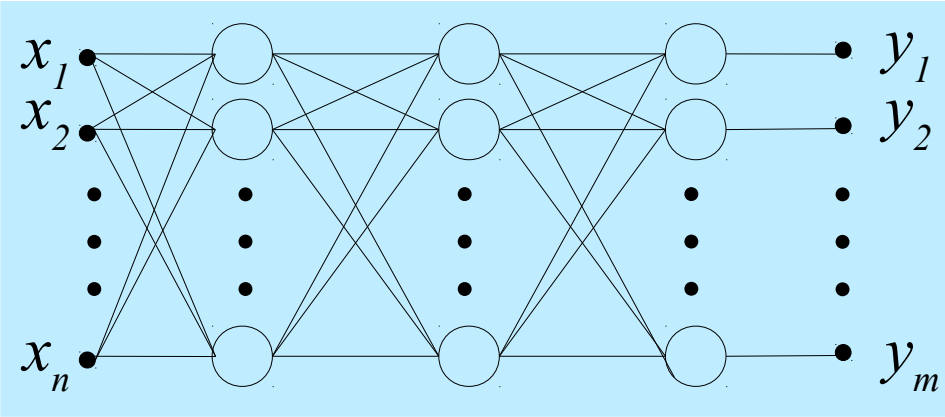
$$y = \sigma(W \cdot x)$$

$$\sigma' \sim y(1-y)$$

$$\Delta w \sim e\, y(1-y)\, x$$

- Die Ableitung der Transferfunktion leistet dort den größten Korrekturbeitrag, wo sich das Netz noch nicht auf einen der Zustände 0 oder 1 eingestellt hat.

Neuronale Netze



- Dienen die Ausgangsignale einer Neuronen Schicht als Eingangssignal für eine weitere Neuronen Schicht, so entsteht ein Neuronales Netz.
- Die Topologie des Netzes wird durch die Gewichte w_{ijk} bestimmt. Ein Gewicht 0 entspricht einer nicht existierenden Verbindung, ein negatives Gewicht bedeutet eine Hemmung des jeweiligen Signal.

NN mit Hidden Layer

- Ausgabefunktion eines zweischichtigen Netzes:

$$y_j = \sigma\left(\sum_k w_{jk}\,\sigma\left(\sum_i \omega_{ki}\,x_i - \theta_k\right) - \theta_j\right)$$

$$y_j = \sigma\left(\sum_k w_{jk}\,\sigma\left(\sum_i \omega_{ki}\,x_i\right)\right)$$

$$\vec{y} = \vec{\sigma}\left(W\,\vec{\sigma}\left(\Omega\,\vec{x}\right)\right) \equiv \vec{\sigma}\left(W\,\vec{h}\right)$$

- Das Ausgabesignal ergibt sich aus der Komposition der Übertragungsfunktion zweier Neuronenschichten.
- Problem: gegen welche Ausgabevektoren sollen die verborgenen Schichten optimiert werden?

Topologisches

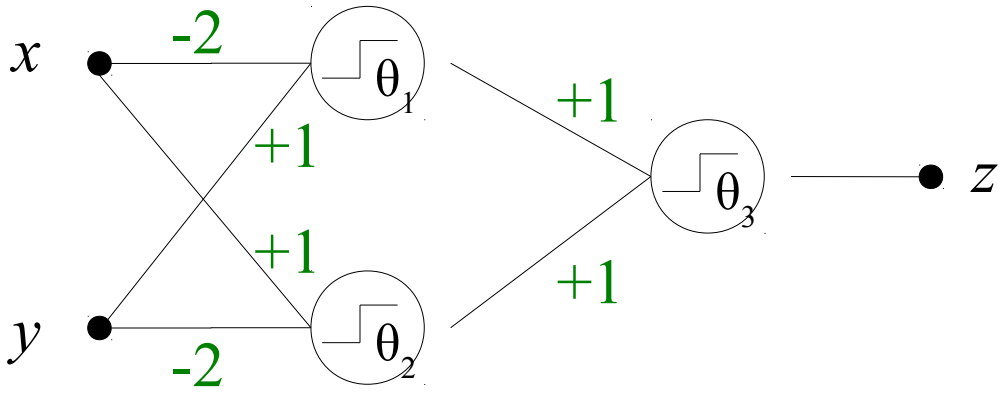
- In dieser einfachen Variante sind die Neuronen nur in Feed-Forward Richtung miteinander vernetzt.
- Eine Rückkoppelung findet weder innerhalb einer Schicht noch zwischen den Schichten statt.
- Da somit jeder Ausgang der vorhergehenden Schicht mit jedem Eingang der nachfolgenden Schicht verbunden ist, gibt es keine „höhere Topologie“. Der Grad der Vernetzung und alle topologischen Informationen sind in der Belegung der Gewichte hinterlegt.
- Regeln, Muster und Gesetzmäßigkeiten werden dem Netz antrainiert und sind nicht eineindeutig ablesbar.

Lösung des XOR Problems

- Aus der Boolschen Algebra ist bekannt, das sich die XOR Funktion durch NOT, AND und OR Gatter erstellen lässt:

$$x \oplus y = (\neg x \wedge y) \vee (x \wedge \neg y)$$

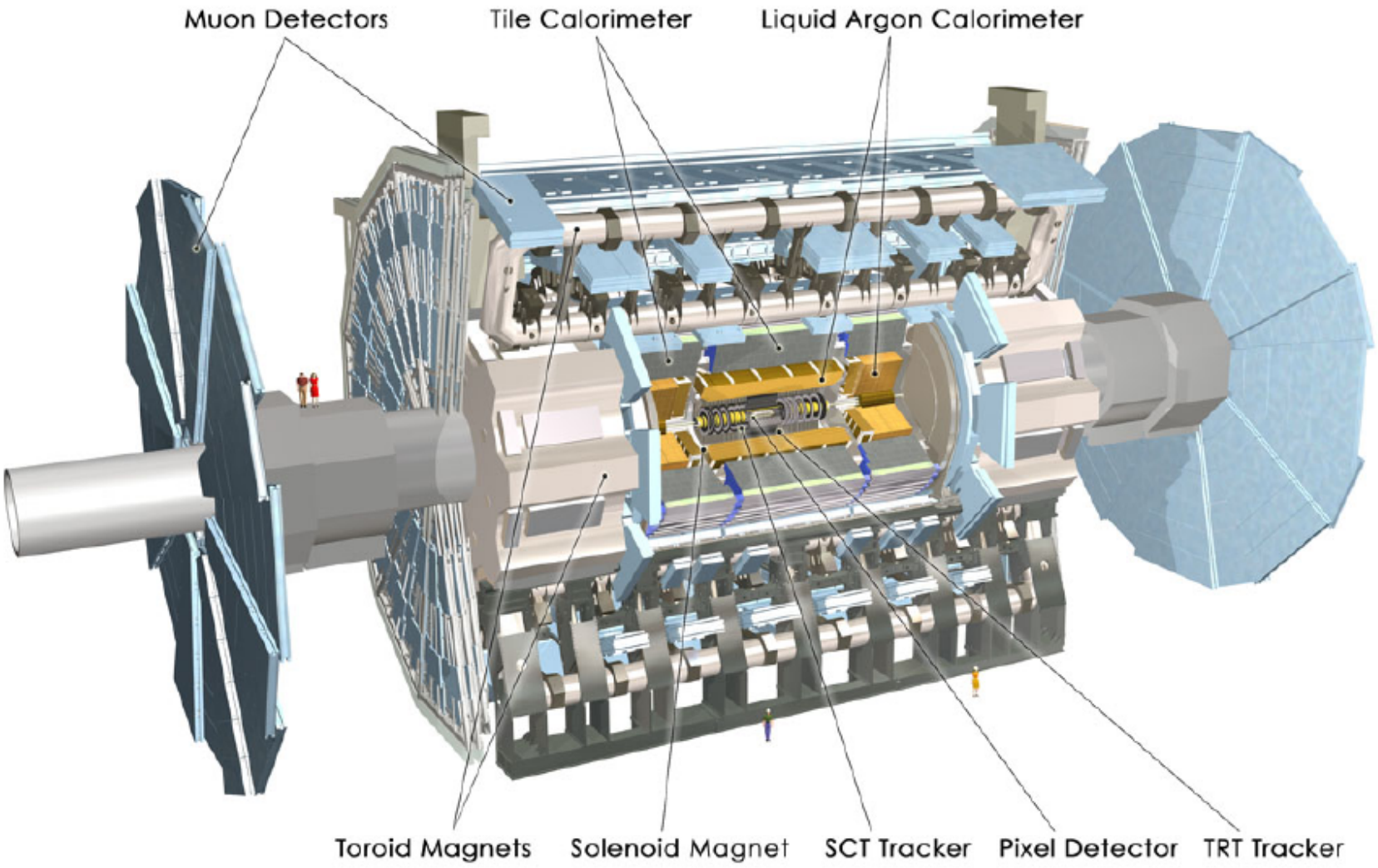
- Da AND und OR mit jeweils einem Neuron darstellbar sind, liegt es nahe für das XOR die folgende Netztopologie mit den Schwellen $\theta = -½$ zu wählen:



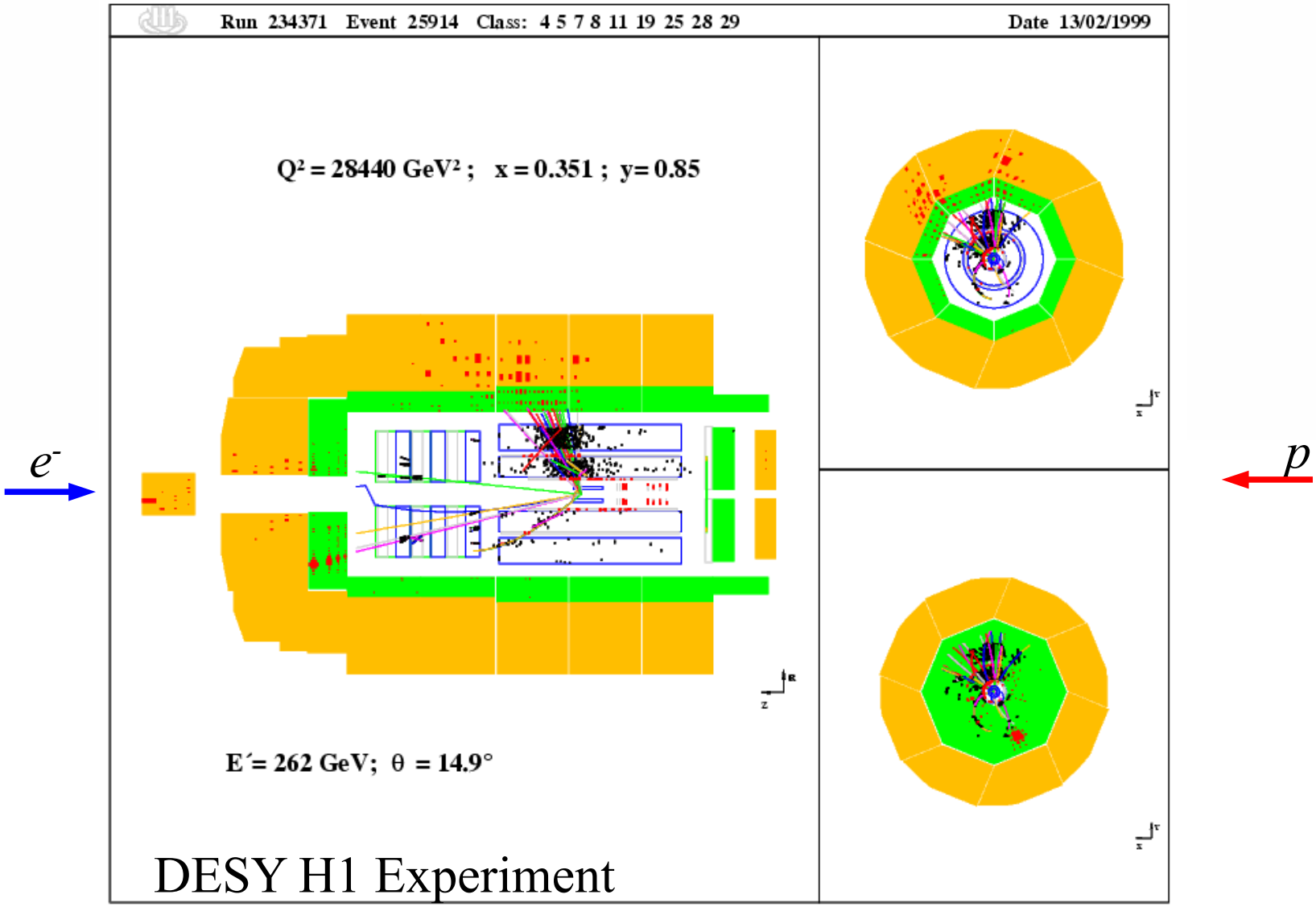
HEP in der Nußschale

- Zur Untersuchung der Struktur der Materie werden Leptonen ($e^{\pm}$ oder $\mu^{\pm}$) oder Nukleon (p oder n) zur Kollision gebracht. Die Leptonen wechselwirken mit den Quarks des Nukleons und dieses zerfällt in seine Bestandteile. Da die Quarks nicht lange bestehen können fragmentieren diese in hadronische Jets, deren Zerfallprodukte als Pionen ($\pi^{\pm}$, π^{0}), Muonen ($\mu^{\pm}$) und Neutrinos (ν) im Detektor nachgewiesen werden.
- Aus der Energie- und Winkelverteilung lassen sich Rückschlüsse auf den Zusammenstoß, eventuelle Resonanzen und auf die Dichteverteilung der Quarks und Gluonen im Nukleon machen.

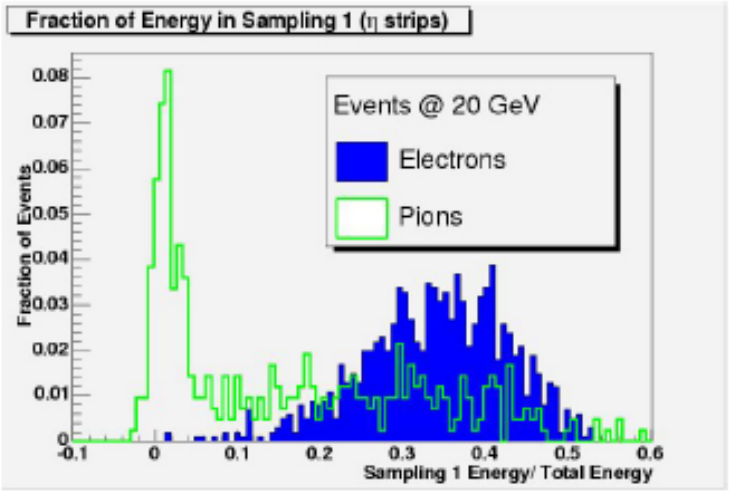
Der ATLAS Detektor

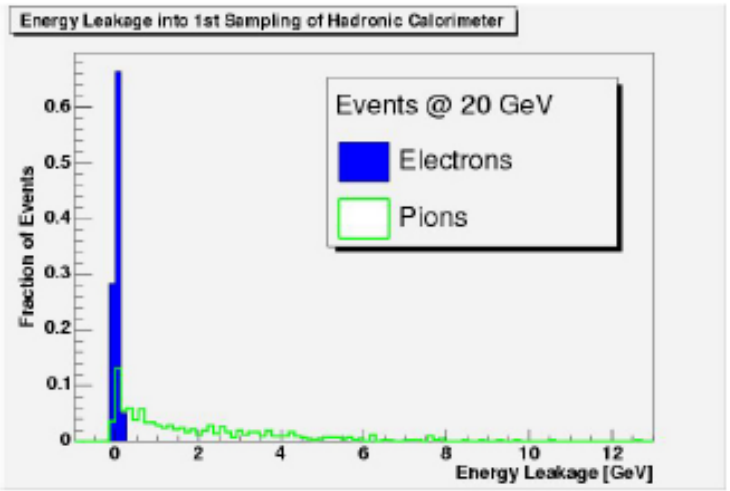


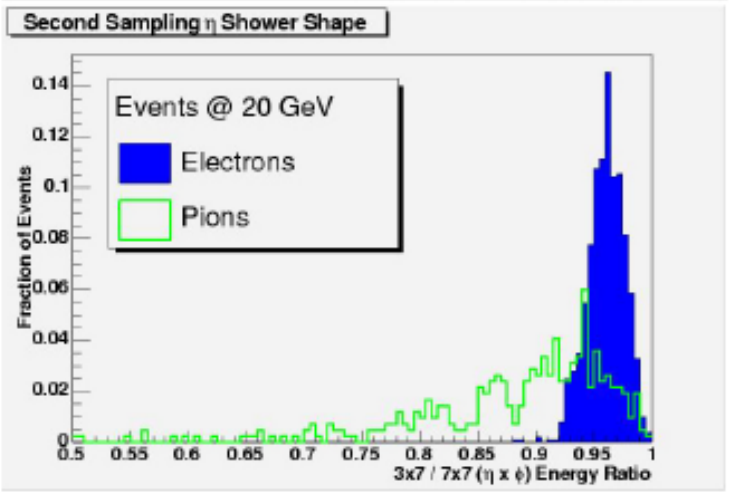
Elektron Proton Streuung

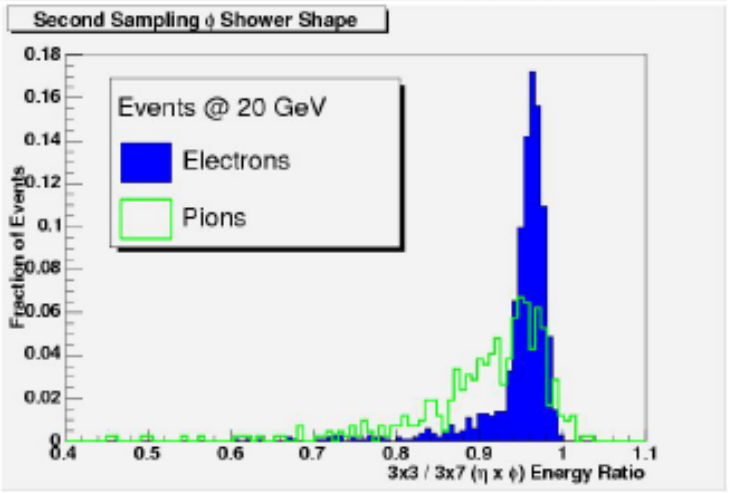


Elektron Pion Verteilungen



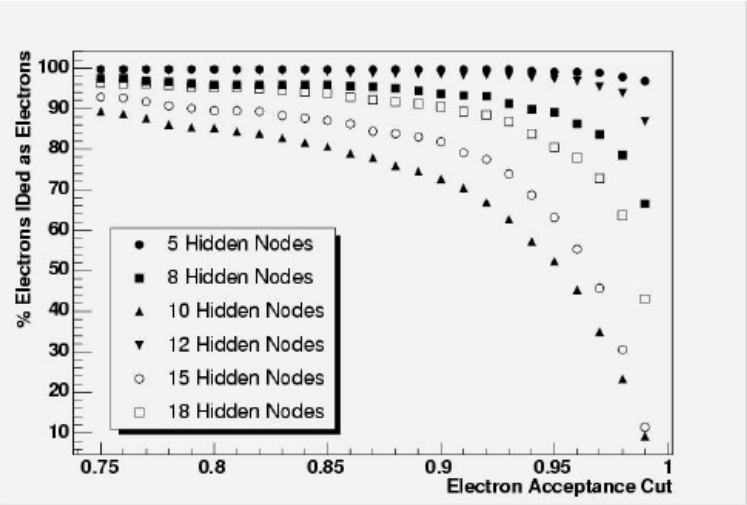


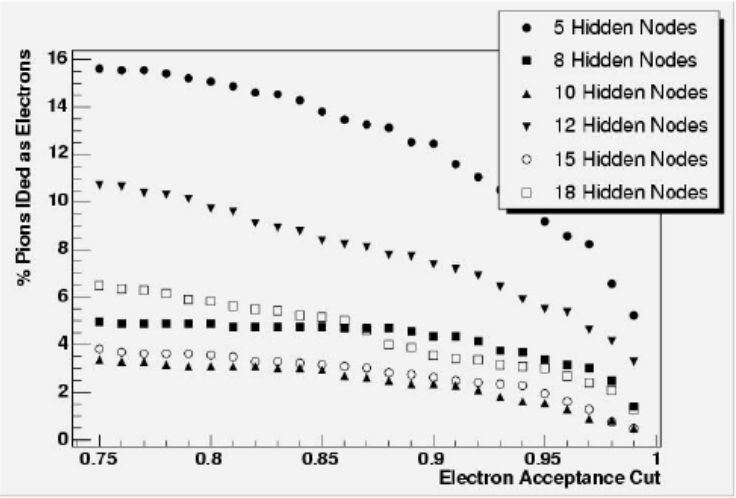




CERN Atlas Simulation

Mustererkennung in HEP





- Wichtig für die richtige Interpretation der Ereignisse ist die Unterscheidung von Elektronen und Pionen.
- Trennung von Elektronen und Pionen mit einen Neuronalen Netz für das ATLAS Experiment am CERN, ist ein höherdimensionales XOR Problem.

Problematisches

- Das XOR Problem konnte durch scharfes Raten gelöst werden, kann ein Netz dies aber auch erlernen?
- Es stellt sich die Frage, wie viele Neuronen und Schichten werden benötigt, um ein gegebenes Problem zu lösen / approximieren?
- Werden zu wenige Neuronen(schichten) gewählt, so reicht die Fähigkeit zum Diskriminieren und zum Erkennen komplexer Zusammenhänge/Muster nicht aus, werden zu viele gewählt so wird die Konvergenz und Lerngeschwindigkeit sehr schlecht sein.
- Mit welchem Kriterium können die Neuronen der Zwischenschicht überhaupt trainiert werden?

Das Kolmogorov Theorem

- Es sei $n>2$ gegeben. Dann existiert eine Familie von reellwertigen, stetig monoton wachsenden Funktionen $h_{ki}:[0,1] \rightarrow [0,1]$, so dass für jede reellwertige stetige Funktion $f:[0,1]^n \rightarrow [0,1]$ eine Familie von stetigen reellwertigen Funktionen $g_k:[0,1]^n \rightarrow [0,1]$ existiert mit:

$$f(\vec{x}) = \sum_{k=1}^{2n+1} g_k \left(\sum_{i=1}^{n} h_{ki}(x_i) \right)$$

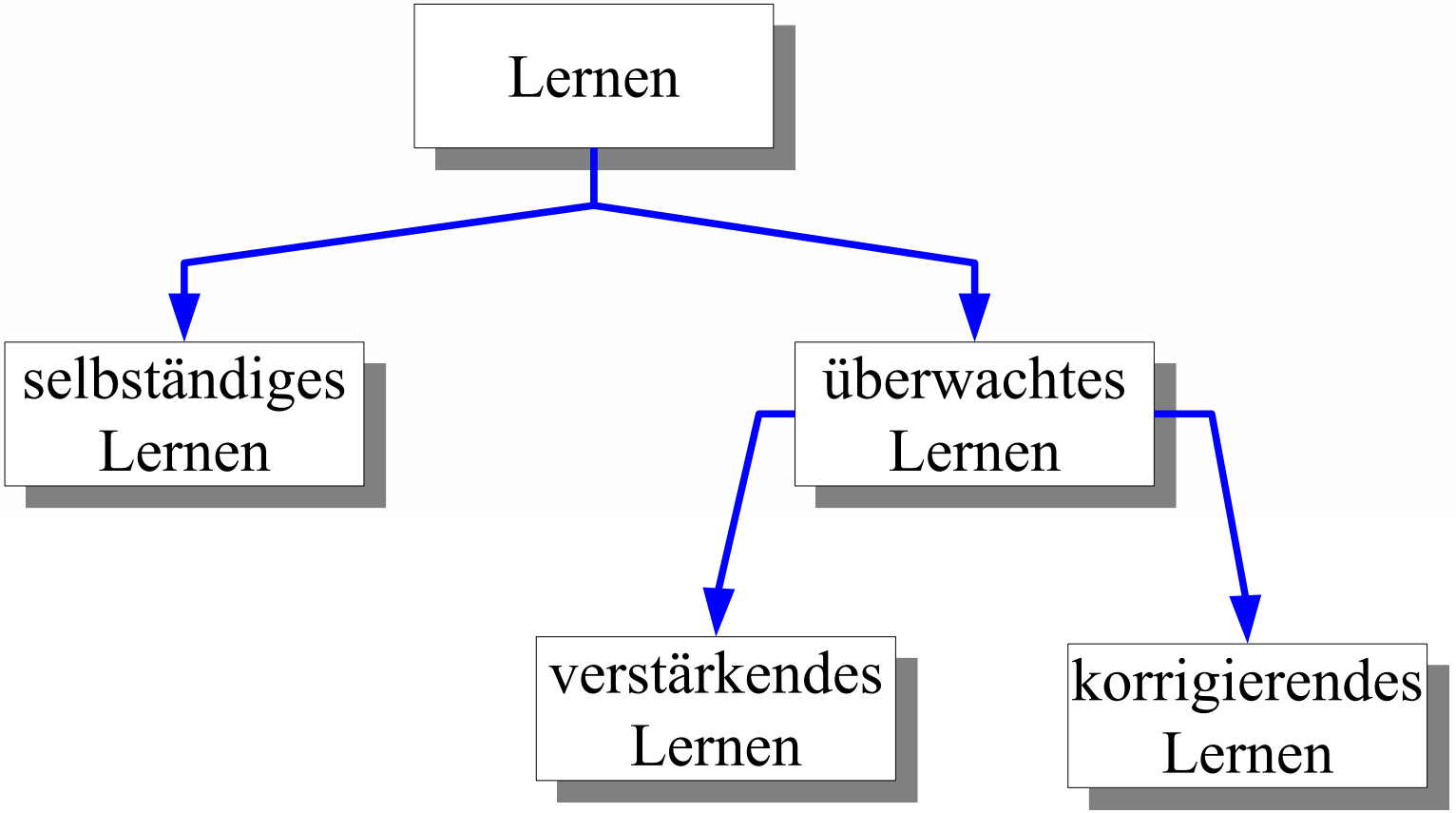
- Kolmogorov (1957):

On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition. AMS Translations, 2(55):55-59.

Bemerkungen zu Kolmogorov

- Neuronales Netz mit *n* Eingängen: Benötigt werden maximal *n(2n+1)* Neuronen in der Zwischenschicht und *2n+1* Neuronen pro Ausgabekanal.
- Das Theorem garantiert die Existenz der Funktionen g_k und h_{ki}, gibt aber kein konstruktives Verfahren zum Auffinden an.
- σ hat die von den h_{ki} auf [0,1] geforderte Eigenschaft der stetigen Monotonie und die Gewichte *w* sind im begrenzten Sinne „anpassungsfähig“.
- Es wird nicht garantiert, dass die Transferfunktion σ immer die gesuchte Funktion g_k oder h_{ki} darstellt.

Einordnung des Lernprozesses



- Überwachtes Lernen erfordert einen Trainer, der entweder „die Wahrheit“ kennt oder aber „belohnt“.

Backward Propagation Algorithmus

- Ein Feedforward-Network ermöglicht keine Rückkoppelung der Neuronensignale untereinander oder vom Ausgang zum Eingang.
- Jedoch in der Trainingsphase werden die Fehler eines mehrschichtigen NN „rückwärts durch das Netz“ vom Ausgang zum Eingang als Korrekturwerte propagiert.
- Diese rückwärts gerichtet Fehlerpropagation war bis 1974 unbekannt, ergab sich zwangsläufig aus der konsequenten Anwendung des Gradientenverfahrens und führte zu einer Renaissance der Neuronalen Netze in den 80er Jahren.

Backward Propagation of Errors

- Ausgehend von der Ausgabefunktion eines zweischichtigen Netzwerks:

$$y_j = \sigma\left(\sum_k w_{jk}\,\sigma\left(\sum_i \omega_{ki}\,x_i\right)\right)$$

- gilt es den quadratischen Fehler zu optimieren:

$$E[W,\Omega] = \frac{1}{2}\sum_{\nu=1}^{p} \left\| y^{(\nu)} - d^{(\nu)} \right\|^2$$

- Der Gradient bezüglich der Ausgabeschicht bleibt unverändert, o.B.d.A wird die Summation über ν unterdrückt:

$$\frac{\partial E}{\partial w_{jk}} = (y_j - d_j)\,y_j(1-y_j)\,h_k$$

h_k: hidden neuron

Fehler der verborgenen Schicht

- h_k ist das Ausgangssignal der vorgelagerten Schicht(en). Die Ableitung nach den Koeffizienten der verborgenen Schicht(en) erfordert die umsichtige Anwendung der Kettenregel:

$$\frac{\partial E}{\partial \omega_{\alpha\beta}} = \sum_k (y_k - d_k)\, y_k (1 - y_k) \sum_j w_{kj} \frac{\partial h_j}{\partial \omega_{\alpha\beta}}$$

$$\frac{\partial E}{\partial \omega_{\alpha\beta}} = \sum_{k,j} (y_k - d_k)\, y_k (1 - y_k)\, w_{kj}\, h_j (1 - h_j) \sum_m \frac{\partial \omega_{jm}}{\partial \omega_{\alpha\beta}} x_m$$

$$\frac{\partial E}{\partial \omega_{\alpha\beta}} = \sum_k (y_k - d_k)\, y_k (1 - y_k)\, w_{k\alpha}\, h_\alpha (1 - h_\alpha)\, x_\beta$$

Verallgemeinerte Lernregel

$$\frac{\partial E}{\partial \omega_{\alpha\beta}} = \sum_k (y_k - d_k)\, y_k (1 - y_k)\, w_{k\alpha}\, h_\alpha (1 - h_\alpha)\, x_\beta$$

$$\frac{\partial E}{\partial \omega_{\alpha\beta}} = h_\alpha (1 - h_\alpha)\, x_\beta \sum_k \underbrace{e_k\, y_k (1 - y_k)}_{\delta_k}\, w_{k\alpha}$$

- Zusammengefasst ergibt dies für alle *i* Schichten die universelle Hebbsche Lernformel:

$$\Delta w_{ijk} = \mu\, \delta_j^i\, x_k$$

- mit

$$\delta_j^i = \begin{cases} y_j (1 - y_j)\, e_j \\ h_j (1 - h_j) \sum_m \delta_m^{i+1}\, w_{i+1\,m\,j} \end{cases}$$

für die letzte Ausgabeschicht

rekursiv für die verborgene(n) Schicht(en)

Interpretation der Lernformel

- Die Lernformel für die Ausgabeschicht besagt, die Änderung ist proportional zur Abweichung vom gewünschten Ergebnis.
- Für die innere Schicht (die inneren Schichten) gilt, die Korrektur ist proportional zur Summe der Fehler der nachfolgenden Schicht. Oder auch jedes Neuron ist verantwortlich für alle Folgefehler.

$$\tilde{e}_j = \sum_m \delta_m w_{mj}$$

- In der verborgenen Schicht gibt es für das Neuron j kein klares Fehlerkriterium. Alle Fehler werden rückwärts durch das Netz propagiert...

Praktikum

- Kämpfen Sie sich durch die Indizes und implementieren Sie den BPA für zwei oder mehrere Schichten.
- Testen Sie Ihre Implementierung, indem Sie ein BPN das XOR Problem erlernen lassen.