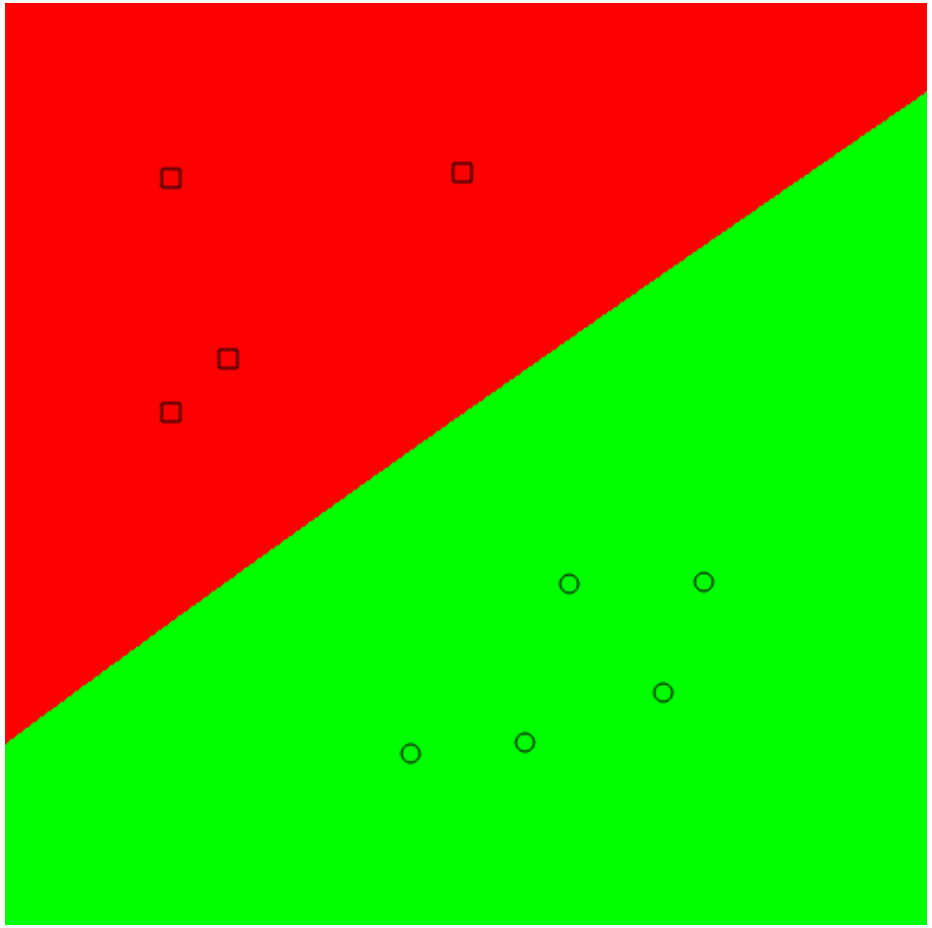
Supervised Learning

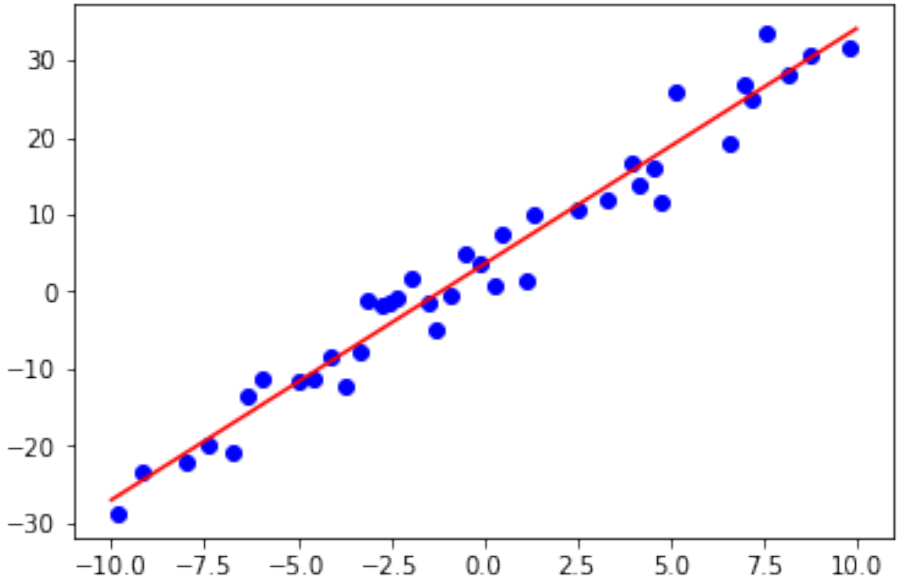
Für Binärklassifizierung

Gliederung

1. Die Problemstellung
2. Adaline (Adaptive Linear Neuron) + Programmierteil
3. (Multilayer Perceptron / Neuronales Netzwerk)

Supervised learning





Die Problemstellung: Binärlassifizierung

Gegeben sind in Vektorform gefasste Eigenschaften eines Objekts oder Erreignis (features):

$$\vec{x}$$

Zu jedem $\vec{x}$ gibt es eine passende Klasse (target):

$$y(\vec{x})$$

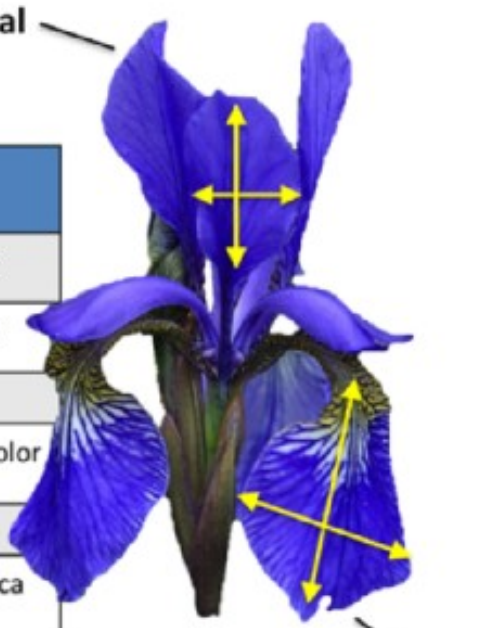
Für Binärklassifizierung gilt:

$$y \in \{0,1\}$$

Beispiel für Binärklassifizierung:

Samples
(instances, observations)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

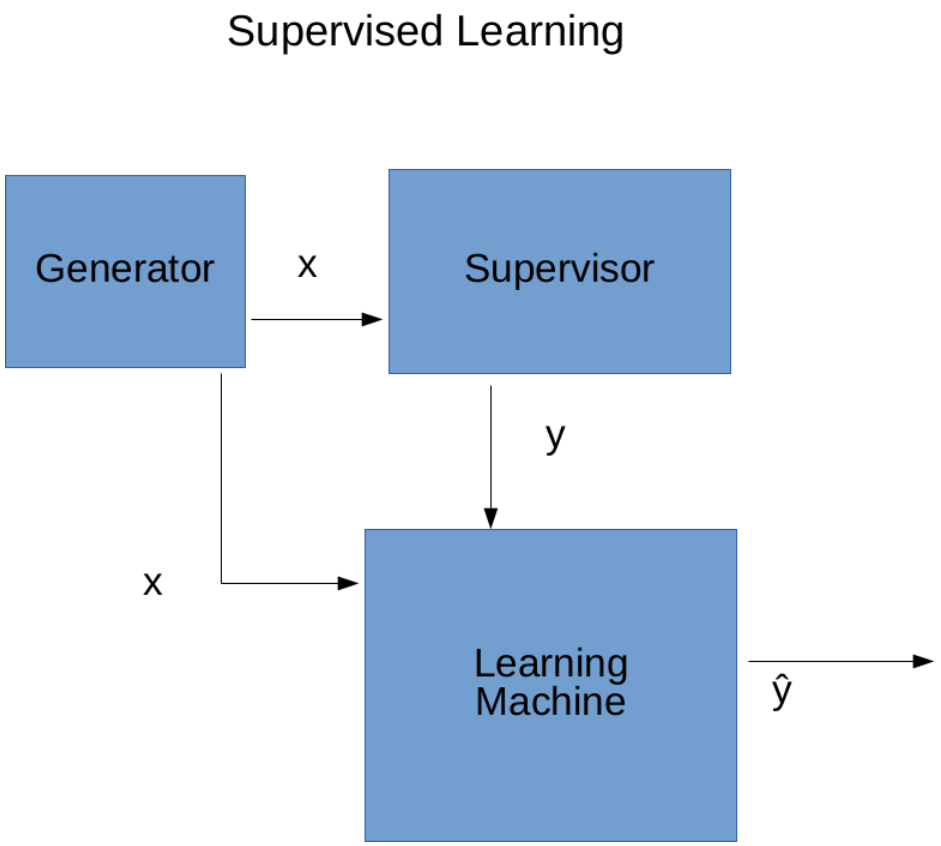


Class labels
(targets)

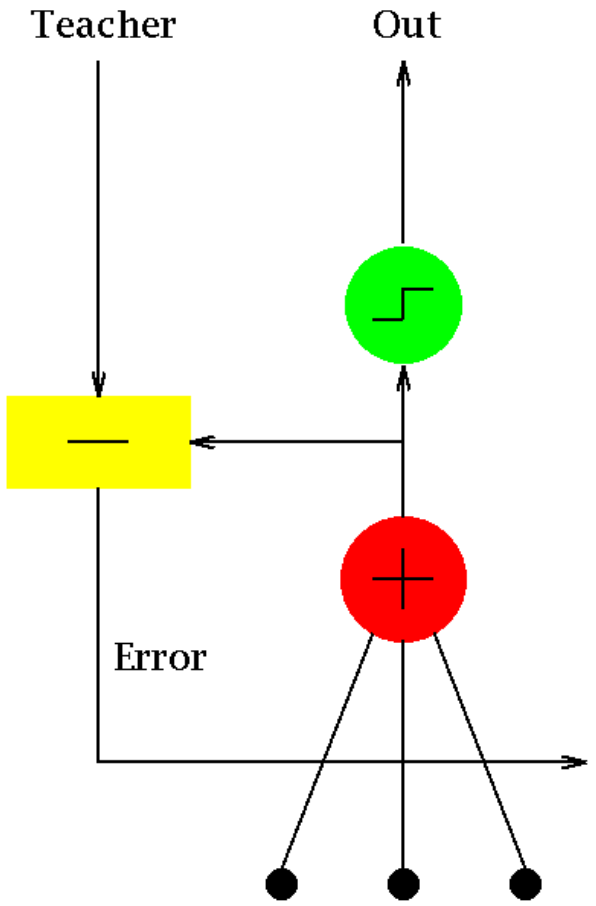
Features
(attributes, measurements, dimensions)

Die Problemstellung: Supervised Learning

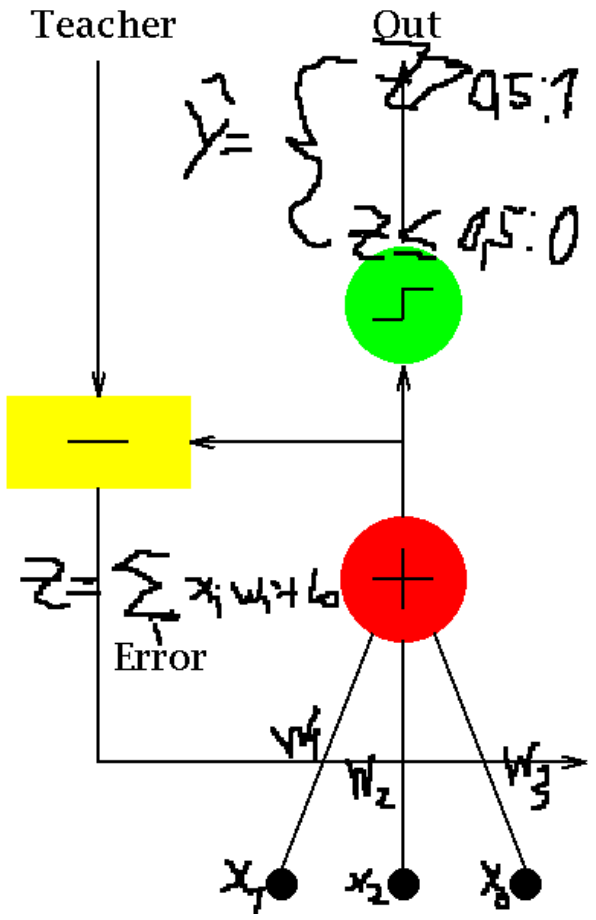
- Gegeben sind Daten x die vom Generator kommen
- Der Supervisor gibt die korrekte Klasse y(x) an die Parameter der Funktion y(x) sind nicht bekannt
- Die Lernmaschine erstellt eine Annäherung $\hat{y}(x)$ die das Verhalten des Supervisors wiedergeben soll.



Adaptive Linear Neuron



Adaptive Linear Neuron



Das Verlustfunktional

- **Um die optimalen Parameter zu finden benötigt man eine Ziel-, oder Verlustfunktion (bzw. ein Verlustfunktional) ,anhand desses die Parameter bewerted werden können:**

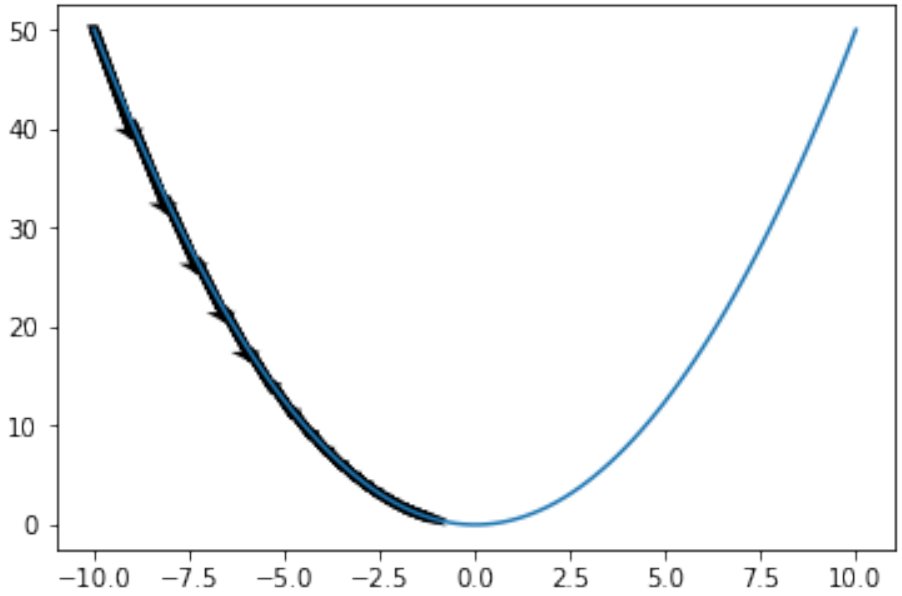
$$z_i = \vec{w} \cdot \vec{x}_i + b$$

$$L(\vec{w}, b) = \frac{1}{2} * \sum_{1}^{l} (y_i - z_i)^2$$

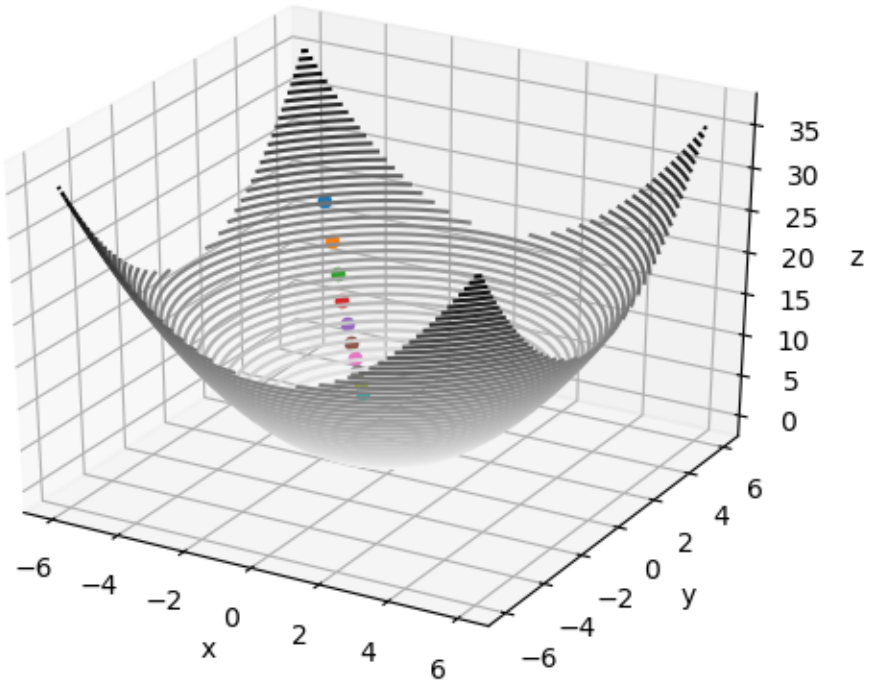
Das Gradientenverfahren: Idee

- **Ableitung = Steigung = Wie Stark geht es aufwärts**

  => Läuft man entgegen der Steigung so landet man irgendwann in einem Minimum



Gradientenverfahren in 3d



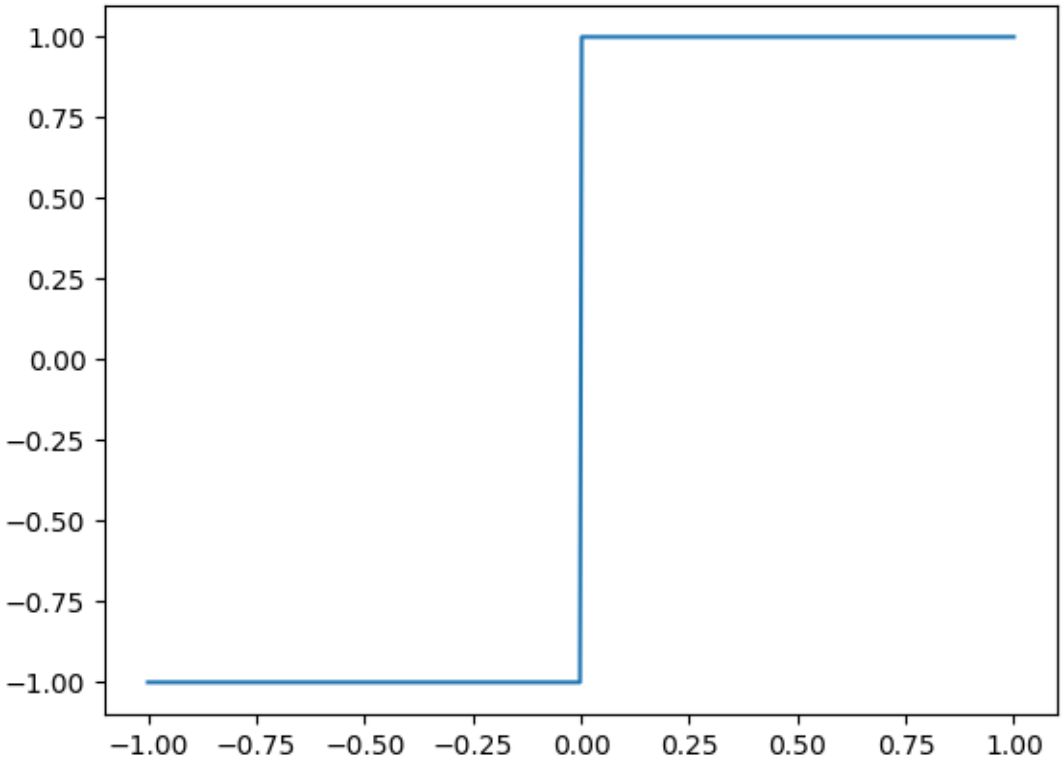
AdalineSGD Pseudocode:

1. **Initialisiere Parameter:** $\vec{w}=\vec{0}, b=0$
2. **Wiederhohle n mal:**
3. **Für alle x_i:**
4. **Für alle w_j, x_i,j:** $w_j := w_j - \alpha * \frac{d L_i}{d w_j}$
5. **Für b:** $b := b - \alpha * \frac{d L_i}{d b}$

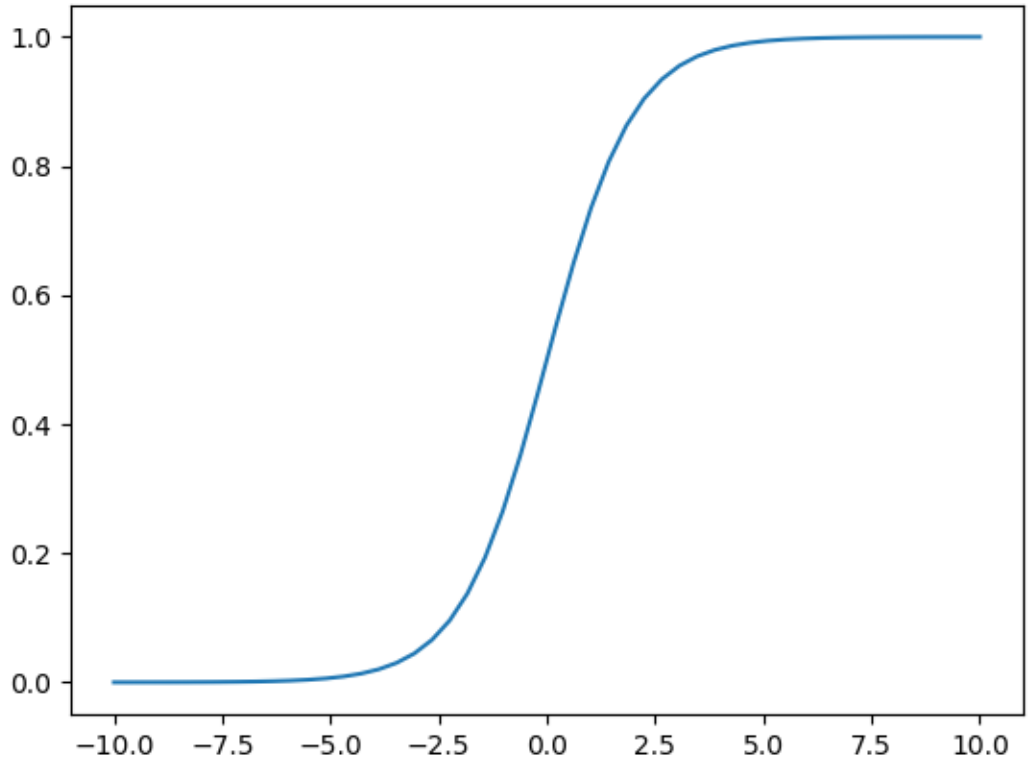
Semantische analyse mit Tfidf

- **Dokumente können dargestellt werden, indem man die Wörter zählt und diese dann als vektor darstellt.**
- **Mit Tfidf wird die Zählung umgewichted, indem Wörter, die in allen dokumenten häufig vorkommen abgewerted werden.**

Diskrete Aktivierungsfunktion (Schrittfunktion)

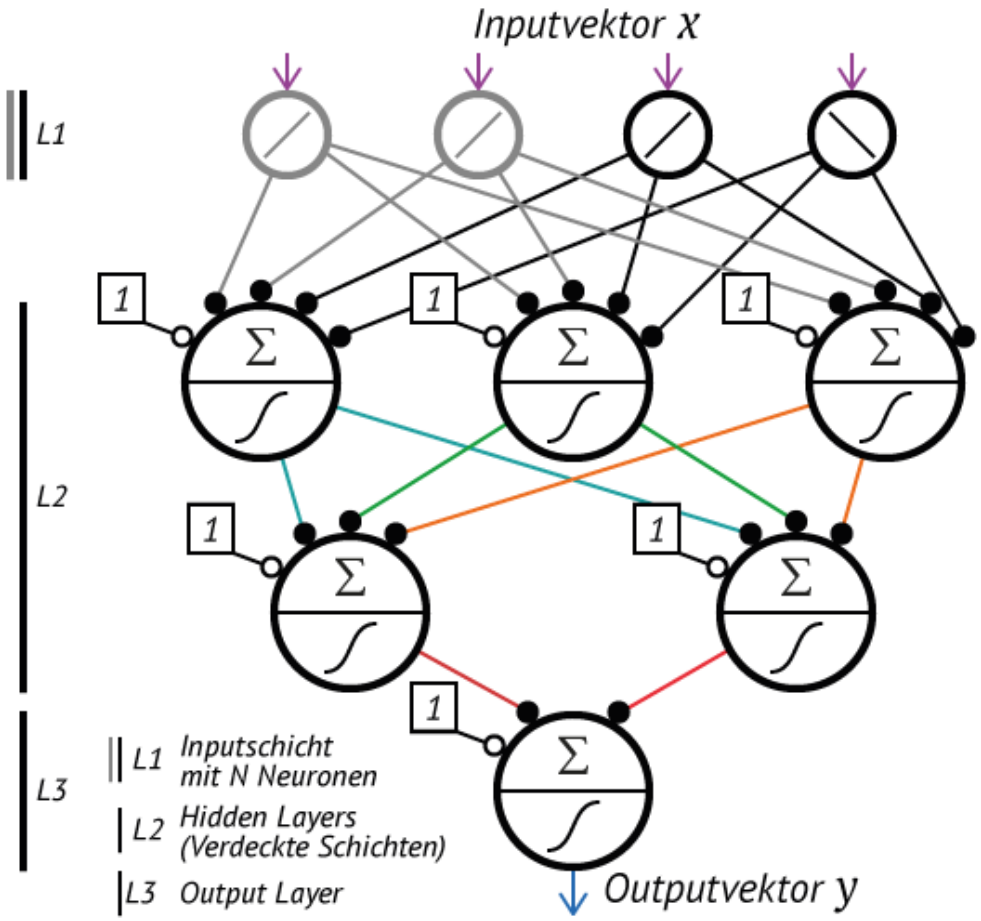


Kontinuirliche Aktivierungsfunktion / Sigmoid Funktion

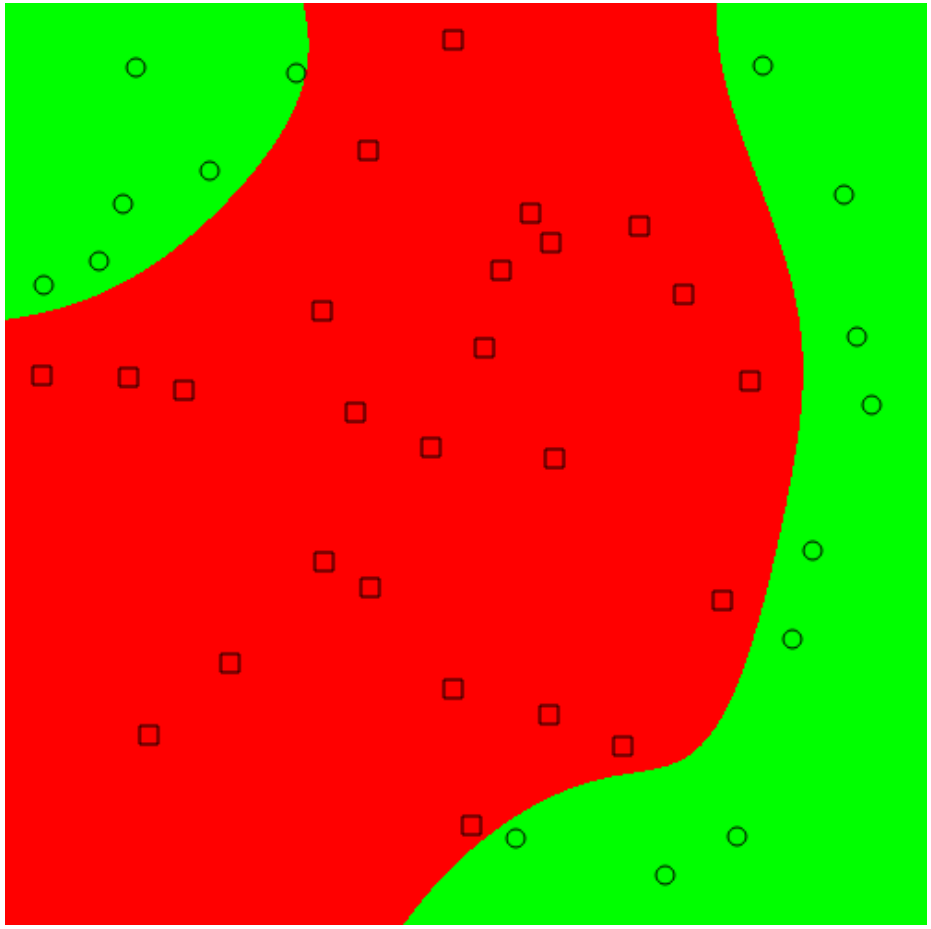


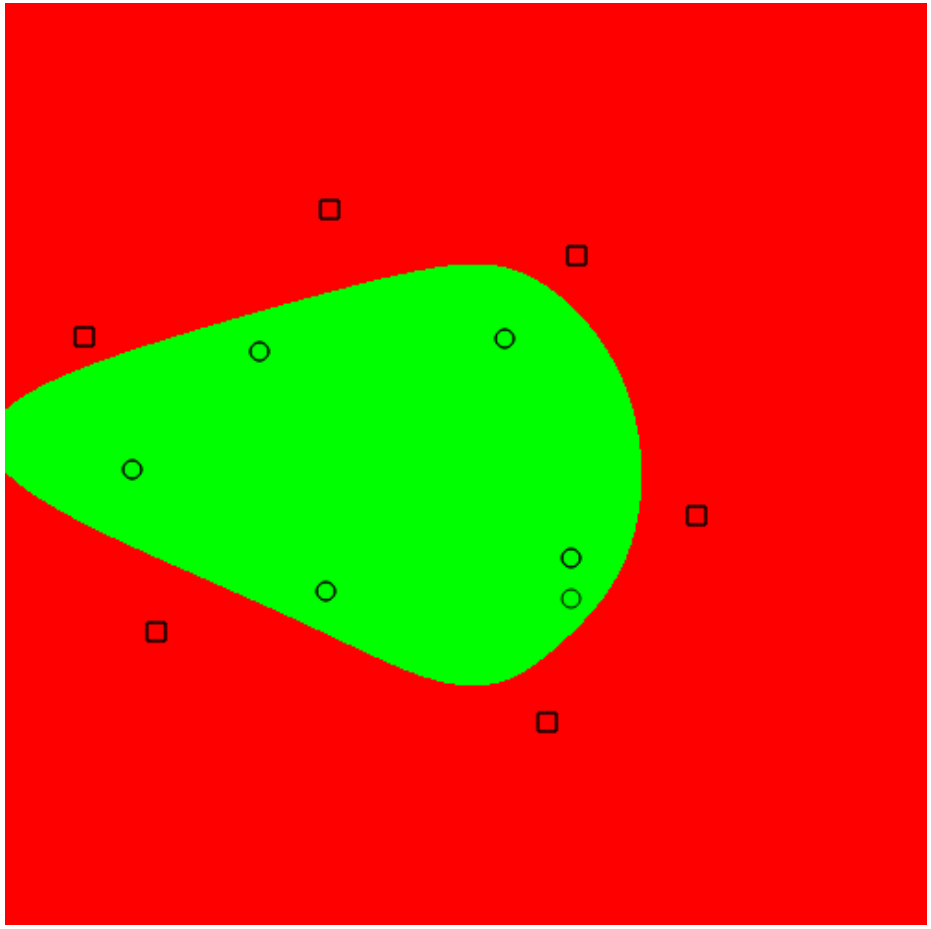
$$\sigma(x) = \frac{1}{(1+e^{-x})}$$

Multilayer-Perzeptron / Neuronales Netzwerk

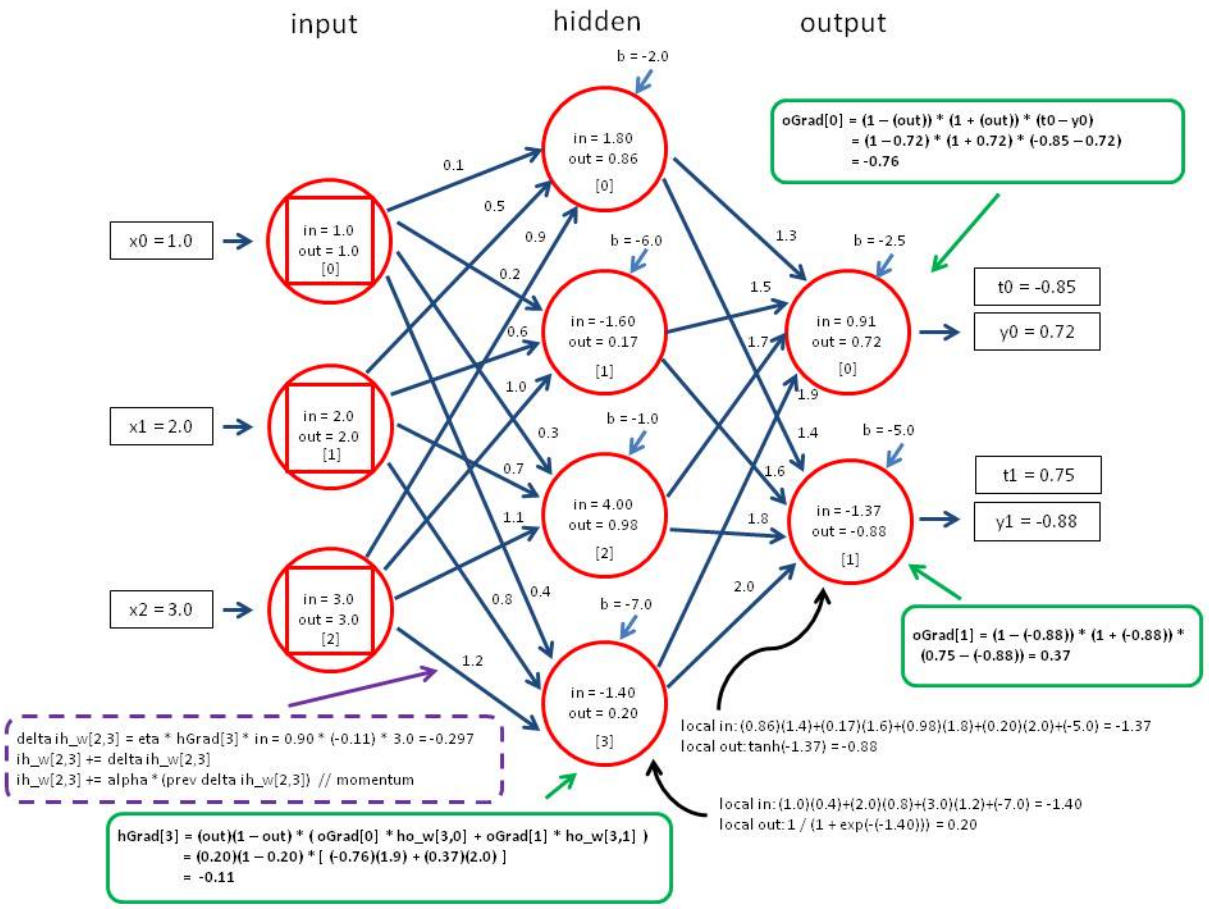


Nichtlineare Klassifizierung mit Neuronalen Netzwerken





Trainingsverfahren Backpropagation



Anwendungen Neuronale Netze

- **Medizin:**
  - Bilderkennung Hautkrebs
- **Autoindustrie:**
  - Bilderkennung und Lokalisation von Objekten für Selbstefahrende Autos
- **Management / Operations Research:**
  - Vorhersage und Analyse von Kundenverhalten
  - Gewinn- / Risikovorhersagen
- **Chemie:**
  - Vorhersage von Molekühleigenschaften

Quellcode:

Quellen

- **Bilder:**
  - http://www.informatikseite.de/neuro/img42.png
  - https://jamesmccaffrey.files.wordpress.com/2012/11/backpropagationcalculations.jpg
  - http://www.bogotobogo.com/python/scikit-learn/images/features/iris-data-set.png
  - https://upload.wikimedia.org/wikipedia/commons/b/be/Adaline_flow_chart.gif
- **Bücher:**
  - Statistical Learning Theory (Vladimir N. Vapnik 1998)
  - Python Machine Learning (Sebastian Raschka 2015)
  - Deep Learning (Ian Goodfellow, Youshua Bengio, Aaron Courville 2016)

Graphiken wurden mit python matplotlib und html5 canvas erstellt.