FRANKFURT UNIVERSITY OF APPLIED SCIENCES

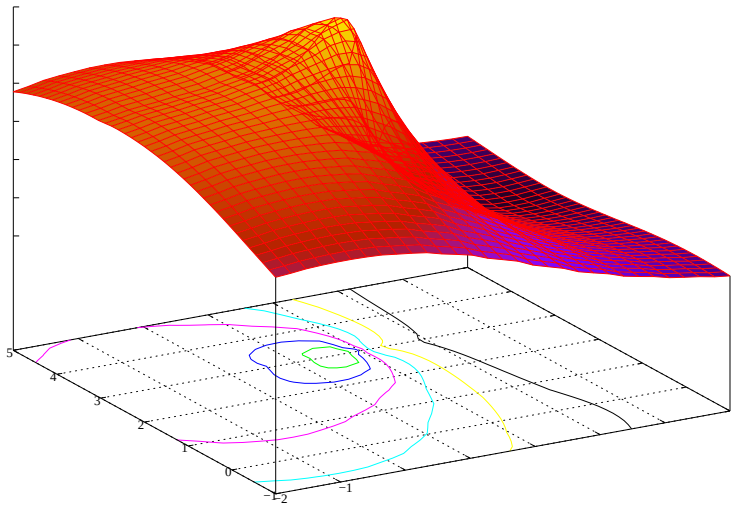
Vorlesung
Grundlagen adaptiver Wissenssysteme

Prof. Dr. Thomas Gabel
Frankfurt University of Applied Sciences
Faculty of Computer Science and Engineering
tgabel@fb2.fra-uas.de

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Vorlesungseinheit 10

Repräsentation der Wertfunktion im Reinforcement Learning



Repräsentation der Wertfunktion im Reinforcement Learning

Lernziele

- Repräsentation der Pfadkosten bei sehr großen / kontinuierlichen Zustandsräumen
- typische Formen der Funktionsapproximation beim Reinforcement Lernen

Schwerpunkte

- Diskretisierung
- Bäume
- Lineare Modelle
- Neuronale Netze

Repräsentation der Wertfunktion im Reinforcement Learning

Überblick

Repräsentation der Wertfunktion im Reinforcement Learning

Überblick

Repräsentation der Pfadkosten $V(\cdot)$

Bis jetzt: Endliche Zustandsmengen $S = \{0, 1, \ldots, n\}$
$\Rightarrow$ V: eindimensionales Array der Länge n
$\Rightarrow$ Q: $n \times m$-dimensionale Tabelle mit m = Anzahl der Aktionen

Repräsentation der Pfadkosten $V(\cdot)$

Bis jetzt: Endliche Zustandsmengen $S = \{0, 1, \ldots, n\}$
$\Rightarrow$ V: eindimensionales Array der Länge n
$\Rightarrow$ Q: $n \times m$-dimensionale Tabelle mit m = Anzahl der Aktionen
Aber: Was, wenn n sehr gross (z.B. Backgammon: 10^{15}) oder S

einen kontinuierliche Wertebereich (Robotik, Regelungstechnik) hat?
$\Rightarrow$ Approximation der Wertfunktion

- Diskretisierung
- Funktionsapproximation durch Regression

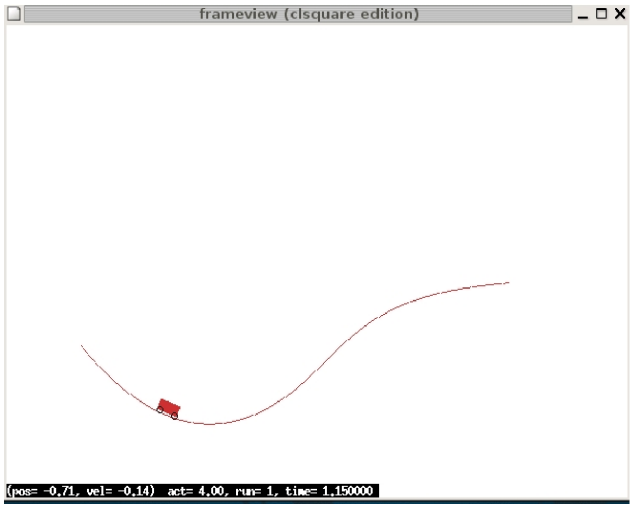
Repräsentation der Wertfunktion im Reinforcement Learning

Überblick

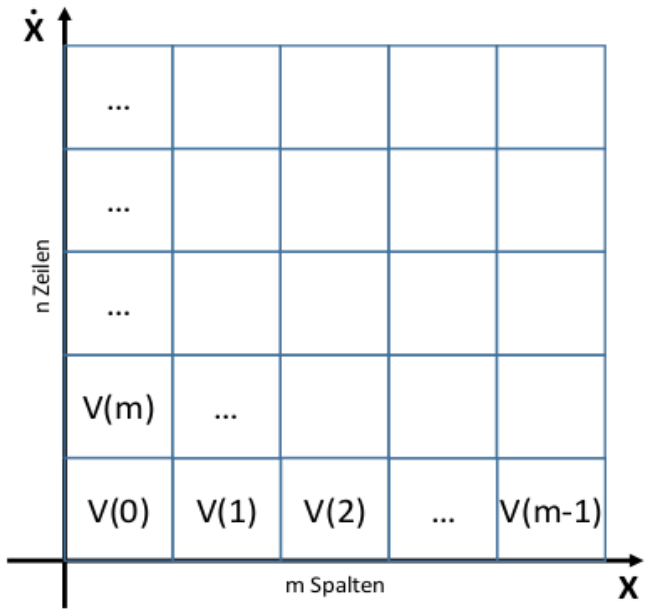
1. Motivation
2. Diskretisierung
3. Regression
4. Bäume
5. Lineare Modelle
6. Basisfunktionen
7. Instanzen- und fallbasierte Methoden
8. Neuronale Netze

Regelmäßige Diskretisierung

MountainCar



Tabelle

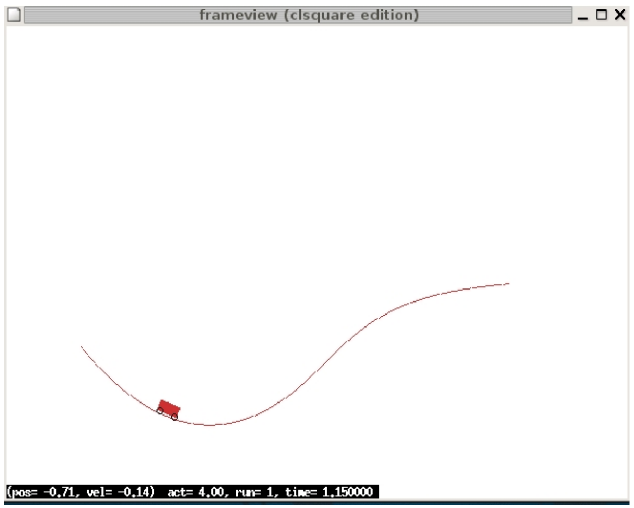


Eingaberaum ist zweidimensional:
x : Position
$\dot{x}$: Geschwindigkeit
Ausgabe: V (Kostenfunktion, Wertfunktion)

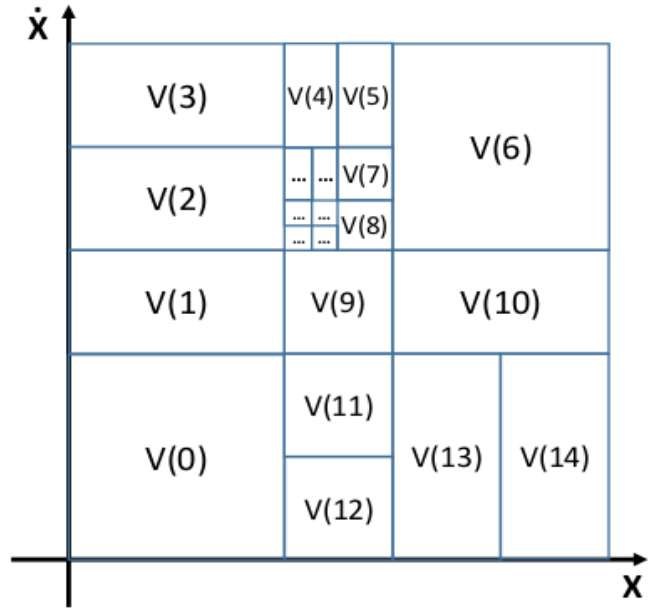
Zustandsraum wird regelmässig diskretisiert mit Auflösung (m, n)
$\Rightarrow$ Pfadkosten sind stückweise konstant

Unregelmäßige Diskretisierung

MountainCar



Tabelle



Eingaberaum ist zweidimensional:
x : Position
$\dot{x}$: Geschwindigkeit
Ausgabe: V (Kostenfunktion, Wertfunktion)

Individuelle Auflösung einzelner Bereiche des Zustandsraumes
$\Rightarrow$ Pfadkosten sind stückweise konstant

Diskussion (Diskretisierung)

Vorteile

- effiziente Datenstruktur (keine aufwendigen Berechnungen nötig)
- transparent (Inspektion der Tabelle möglich)
- theoretisch gut analysierbar

Diskussion (Diskretisierung)

Vorteile

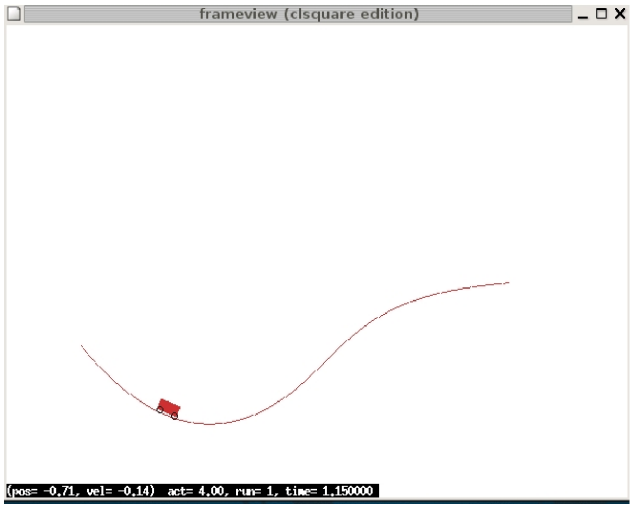
- effiziente Datenstruktur (keine aufwendigen Berechnungen nötig)
- transparent (Inspektion der Tabelle möglich)
- theoretisch gut analysierbar

Nachteile

- hoher Speicherplatzbedarf (“Fluch der Dimensionen”, “Curse of Dimensionality”)
- geringe Generalisierungsleistung
- für praktische Probleme kaum einsetzbar

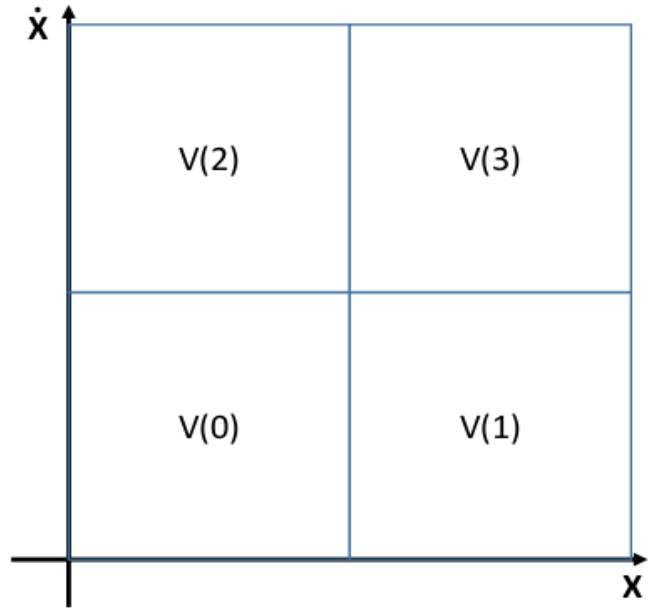
Adaptive Diskretisierungen (1)

MountainCar



Eingaberaum ist zweidimensional:
x : Position
$\dot{x}$: Geschwindigkeit
Ausgabe: V (Kostenfunktion, Wertfunktion)

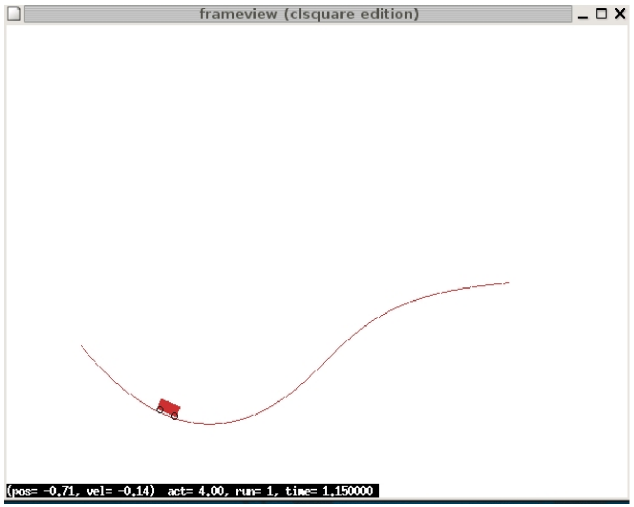
Tabelle



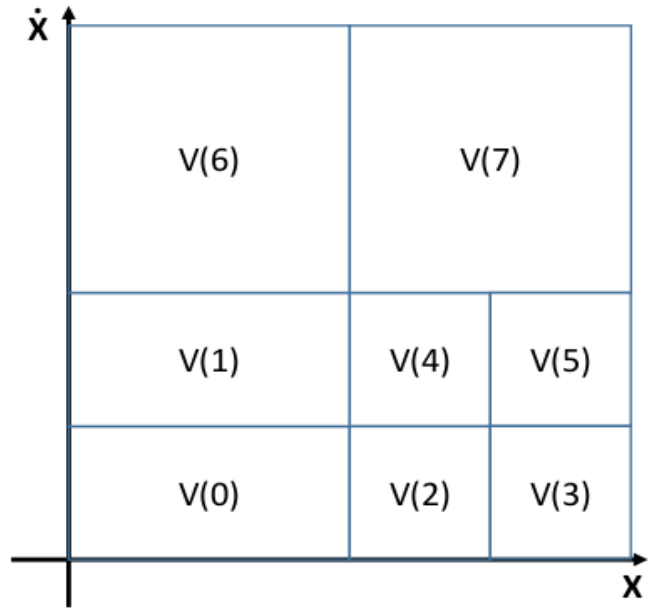
Beginn mit sehr grober Diskretisierung ...

Adaptive Diskretisierungen (2)

MountainCar



Tabelle

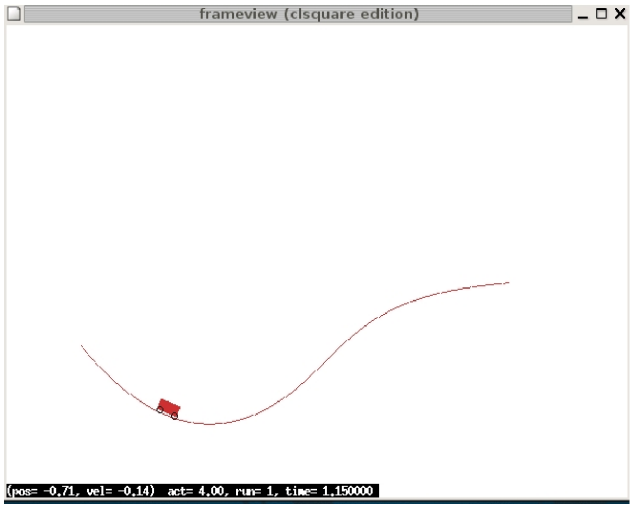


Eingaberaum ist zweidimensional:
x : Position
$\dot{x}$: Geschwindigkeit
Ausgabe: V (Kostenfunktion, Wertfunktion)

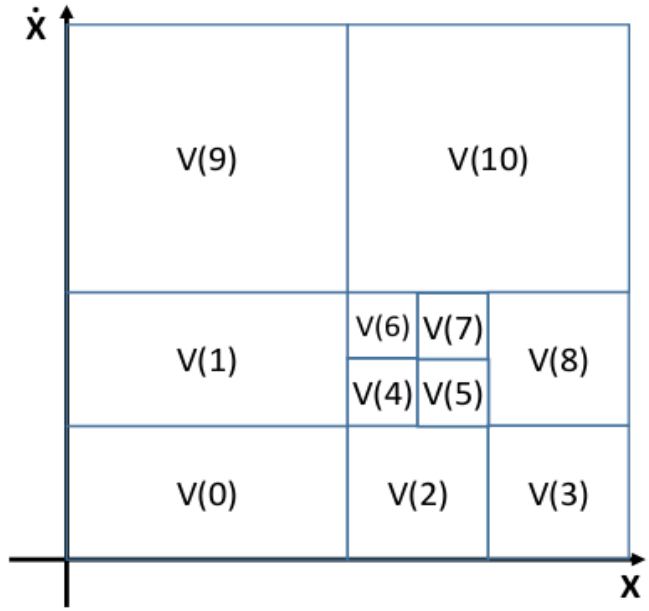
... Verfeinerung nur in wichtigen Bereichen des Zustandsraumes

Adaptive Diskretisierungen (3)

MountainCar



Tabelle



Eingaberaum ist zweidimensional:
x : Position
$\dot{x}$: Geschwindigkeit
Ausgabe: V (Kostenfunktion, Wertfunktion)

Problem: Wann ist ein Zustandsbereich wichtig?

Repräsentation der Wertfunktion im Reinforcement Learning

Überblick

1. Motivation
2. Diskretisierung
3. Regression
4. Bäume
5. Lineare Modelle
6. Basisfunktionen
7. Instanzen- und fallbasierte Methoden
8. Neuronale Netze

Regression

Idee: Approximation von Datenpunkten durch Schätzen eines funktionalen Zusammenhangs

Regression

Idee: Approximation von Datenpunkten durch Schätzen eines funktionalen Zusammenhangs

Gegeben: Mustermenge

$$D = \{(\mathbf{x}^1, \mathbf{t}^1), (\mathbf{x}^2, \mathbf{t}^2), \ldots, (\mathbf{x}^P, \mathbf{t}^P)\}$$

mit

$$\mathbf{x} \in \mathbb{R}^n, \mathbf{t} \in \mathbb{R}^m$$

Regression

Idee: Approximation von Datenpunkten durch Schätzen eines funktionalen Zusammenhangs

Gegeben: Mustermenge

$$D = \{(\mathbf{x}^1, \mathbf{t}^1), (\mathbf{x}^2, \mathbf{t}^2), \ldots, (\mathbf{x}^P, \mathbf{t}^P)\}$$

mit

$$\mathbf{x} \in \mathbb{R}^n, \mathbf{t} \in \mathbb{R}^m$$

Annahme: $\mathbf{t}^i = f(\mathbf{x}^i) + \eta$, mit η Zufallsvariable, "Rauschen"

Gesucht: "Modell", d.h. eine Funktion $\mathbf{y}(\mathbf{x}, \mathbf{w})$, die $f(.)$ möglichst genau approximiert. Dabei ist $\mathbf{w}$ der Vektor der Parameter des Modells.

Regression

Beispiel:
Eingaberaum eindimensional, gesuchtes Modell sei eine Gerade

$$\mathbf{x} = x_1; \mathbf{w} = (a, b)$$

$$y(\mathbf{x}, \mathbf{w}) = a\,x_1 + b$$

Allgemein:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1\,x_1 + w_2\,x_2 + \ldots w_n\,x_n$$

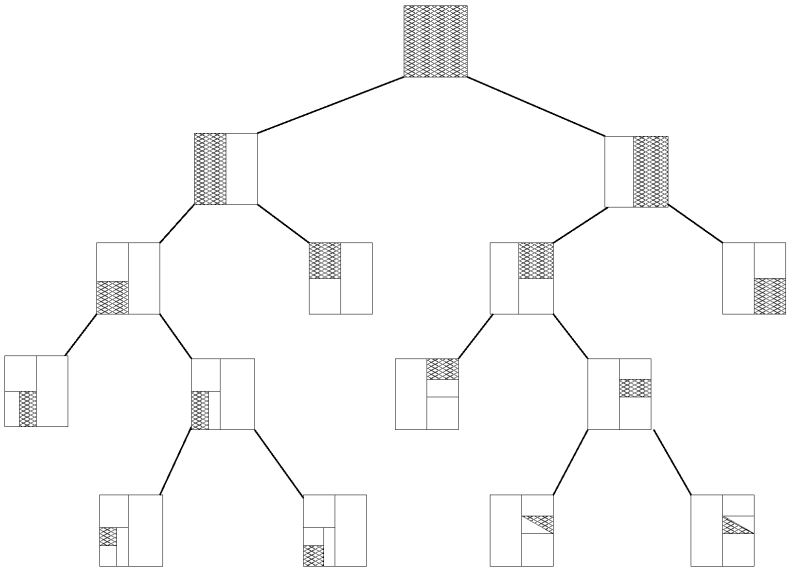
Repräsentation der Wertfunktion im Reinforcement Learning

Überblick

1. Motivation
2. Diskretisierung
3. Regression
4. Bäume
5. Lineare Modelle
6. Basisfunktionen
7. Instanzen- und fallbasierte Methoden
8. Neuronale Netze

Regressionsbäume

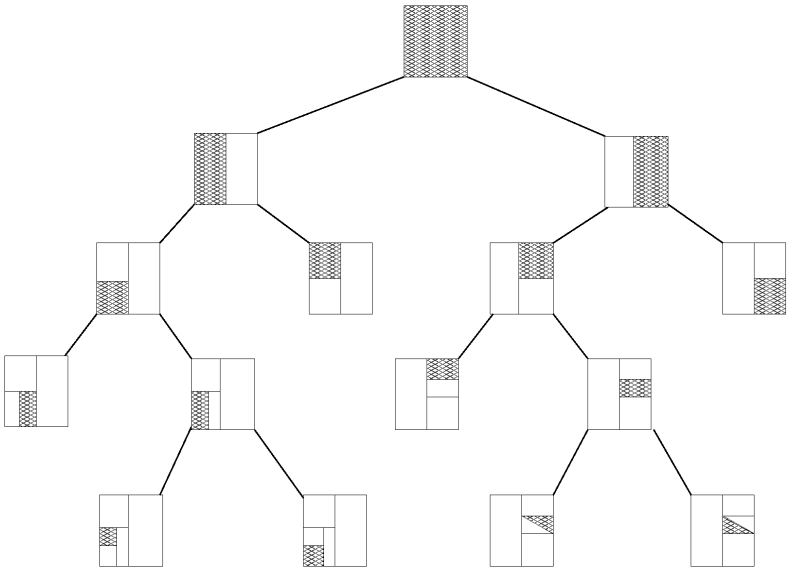
kd-Baum



Blätter werden mit Zielwerten verknüpft
$\Rightarrow$ Diskretisierung mit Baumstruktur

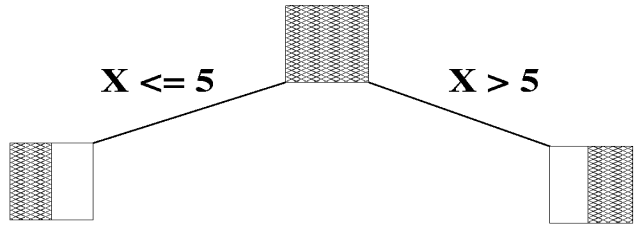
Regressionsbäume

kd-Baum



Blätter werden mit Zielwerten verknüpft
$\Rightarrow$ Diskretisierung mit Baumstruktur

Lernen eines kd-Baumes



- Start mit Wurzelknoten
- Sukzessives Aufspalten der Blätter entlang von Dimensionen des Eingaberaumes
- Wann wird ein Blatt aufgespalten? (z.B. Varianz)

Diskussion (Regressionsbäume)

Vorteile

- adaptiert auf Komplexität des Problems
- informativ (Baumstruktur interpretierbar)
- funktioniert gut mit Ensemble-Techniken
- leistungsfähig auch für schwierige Probleme

Diskussion (Regressionsbäume)

Vorteile

- adaptiert auf Komplexität des Problems
- informativ (Baumstruktur interpretierbar)
- funktioniert gut mit Ensemble-Techniken
- leistungsfähig auch für schwierige Probleme

Nachteile

- gute Splitting-Kriterien schwer zu finden
- tiefe der Bäume ist unbeschränkt
- Ist auch nur eine Form der Diskretisierung ...

Repräsentation der Wertfunktion im Reinforcement Learning

Überblick

1. Motivation
2. Diskretisierung
3. Regression
4. Bäume
5. Lineare Modelle
6. Basisfunktionen
7. Instanzen- und fallbasierte Methoden
8. Neuronale Netze

Lineares Modell

$$
\begin{aligned}
y(\mathbf{x}, \mathbf{w}) &= w_0 + \sum_{i=1}^{n} w_i x_i \\
&= \mathbf{w}^T \mathbf{x}
\end{aligned}
$$

mit $\mathbf{x} = (1, x_1, x_2, \ldots, x_n)^T$, $\mathbf{w} = (w_0, w_1, w_2, \ldots, w_n)^T$

Lineares Modell

$$
\begin{aligned}
y(\mathbf{x}, \mathbf{w}) &= w_0 + \sum_{i=1}^{n} w_i x_i \\
&= \mathbf{w}^T \mathbf{x}
\end{aligned}
$$

mit $\mathbf{x} = (1, x_1, x_2, \ldots, x_n)^T$, $\mathbf{w} = (w_0, w_1, w_2, \ldots, w_n)^T$

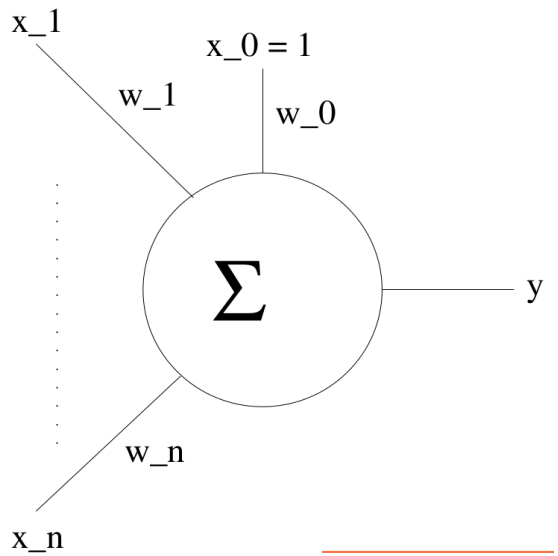
“Lernen”: Suche “möglichst guten” Parametervektor, um die Daten zu beschreiben

Lineares Modell

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^{n} w_i x_i = \mathbf{w}^T \mathbf{x}$$

mit $\mathbf{x} = (1, x_1, x_2, \ldots, x_n)^T$, $\mathbf{w} = (w_0, w_1, w_2, \ldots, w_n)^T$

Modell:



“Lernen”: Suche “möglichst guten” Parametervektor, um die Daten zu beschreiben

⇒ Methode der kleinsten Quadrate – Least-Squares-Methode

Lernen mit Least Squares

Fehler pro Muster: $e^p(\mathbf{w}) = (y(\mathbf{x}^p, \mathbf{w}) - t^p)^2$

Gesamtfehler über alle Muster (Summenbildung):

$$E(\mathbf{w}) := \frac{1}{2}\sum_{p=1}^{P} e^p(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{P}(y(\mathbf{x}^p, \mathbf{w}) - t^p)^2$$

Lernen mit Least Squares

Fehler pro Muster: $e^p(\mathbf{w}) = (y(\mathbf{x}^p, \mathbf{w}) - t^p)^2$

Gesamtfehler über alle Muster (Summenbildung):

$$E(\mathbf{w}) := \frac{1}{2}\sum_{p=1}^{P} e^p(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{P}(y(\mathbf{x}^p, \mathbf{w}) - t^p)^2$$

Gesucht ist $\mathbf{w}^*$ mit

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{w}) = \arg\min_{\mathbf{w}} \frac{1}{2}\sum_{p=1}^{P}(y(\mathbf{x}^p, \mathbf{w}) - t^p)^2$$

Informell: Suche denjenigen Vektor $\mathbf{w}$, der den Abstand der Ausgabe zu den Zielwerten minimiert.

Lernen mit Least Square

Bestimmung von w:
Für das lineare Modell gibt es eine eindeutige Lösung, die man direkt berechnen kann.
Ansatz: Nullsetzen der Ableitung (notwendige Bedingung für ein Minimum (in diesem Fall auch hinreichend)), ergibt schließlich:

$$\mathbf{w}^* := (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \vec{t}$$

mit

$$\mathbf{X} = \begin{pmatrix} (\mathbf{x}^1)^T \\ \vdots \\ (\mathbf{x}^p)^T \end{pmatrix} \tag{1}$$

und $\vec{t} = (t^{(1)}, \ldots, t^{(p)})^T$

Repräsentation der Wertfunktion im Reinforcement Learning

Überblick

1. Motivation
2. Diskretisierung
3. Regression
4. Bäume
5. Lineare Modelle
6. Basisfunktionen
7. Instanzen- und fallbasierte Methoden
8. Neuronale Netze

Basisfunktionen (Features)

Transformation der Eingabedaten

- Eingaben: $\mathbf{x} \in \mathbb{R}^n$
- m nichtlineare Basisfunktionen: $\phi_i : \mathbb{R}^n \rightarrow \mathbb{R}$
- Merkmalsvektor (Feature-Vektor): $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_m(\mathbf{x}))$
- Lineares Modell: $y(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{m} \phi_i(\mathbf{x})\mathbf{w}_i = \mathbf{w}^T \Phi(\mathbf{x})$

Basisfunktionen (Features)

Transformation der Eingabedaten

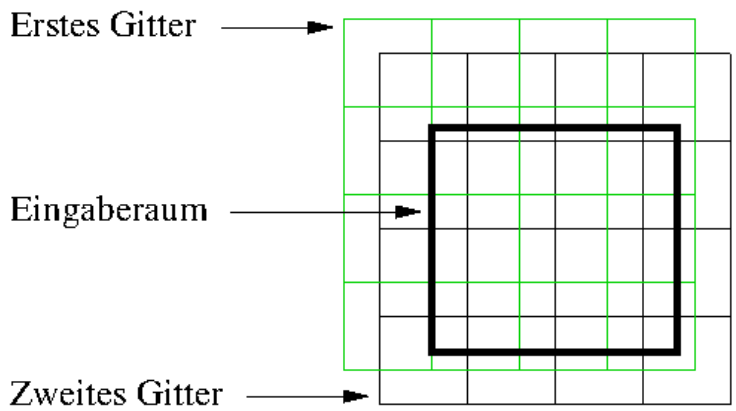
- Eingaben: $\mathbf{x} \in \mathbb{R}^n$
- m nichtlineare Basisfunktionen: $\phi_i : \mathbb{R}^n \rightarrow \mathbb{R}$
- Merkmalsvektor (Feature-Vektor): $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_m(\mathbf{x}))$
- Lineares Modell: $y(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{m} \phi_i(\mathbf{x})\mathbf{w}_i = \mathbf{w}^T \Phi(\mathbf{x})$

Vorteile

- Darstellung von nichtlinearen Zusammenhängen durch lineares Modell
- Extraktion von Zusammenhängen in Eingabedaten
- Einbringung von Vorwissen

Cerebellar Model Articulation Controller (CMAC, 1)

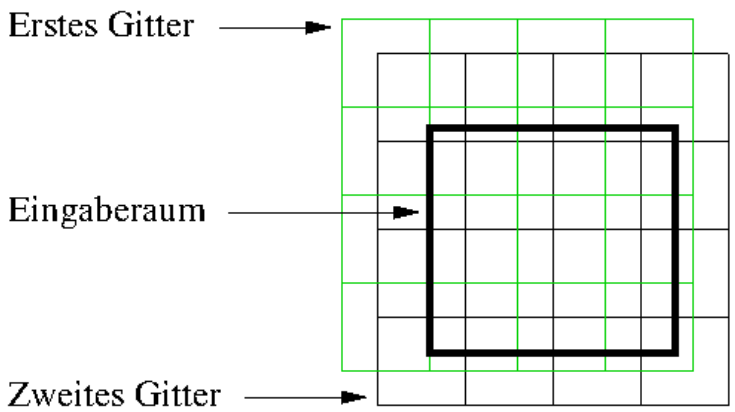
auch bekannt unter “Tile Coding”



Eigenschaften

- stark vereinfachtes Modell eines neuronalen Netzes inspiriert durch ein Modell des Kleinhirns bei Säugetieren
- als Funktionsapproximator erstmalig vorgeschlagen durch James Albus (1975)
- berechnet eine Funktion $f(x_1, \ldots, x_n)$ mit n als Anzahl der Dimensionen
- der n-dimensionale Eingaberaum wird in Hyper-Rechtecke zerlegt (in 2D: Rechtecke), jedes davon wird mit einer Speicherzelle assoziiert
- Inhalte dieser Speicherzellen = Gewichte (also das, was gelernt bzw. angepasst werden soll)

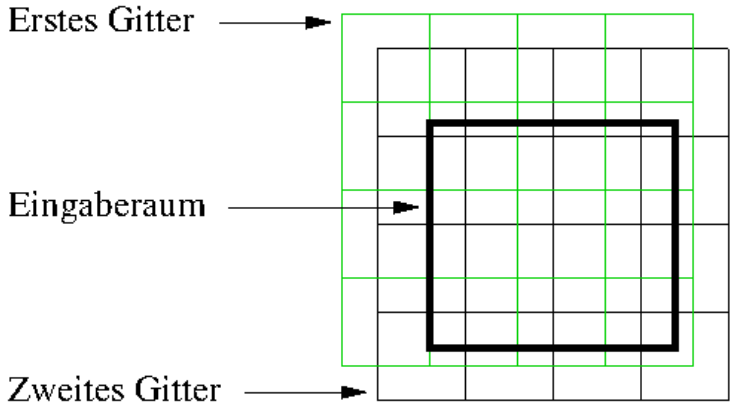
Cerebellar Model Articulation Controller (CMAC, 2)



Besonderheiten

- mehr als eine Diskretisierung des Eingaberaumes wird verwendet
- typisch: mehrere (*m*), leicht zueinander versetzte Gitter
- Ergebnis: jeder Punkt wird mit *m* Hyper-Rechtecke / Speicherzellen assoziiert
- Ausgabe des CMAC ist die arithmetische Summe der Gewichte der Speicherzellen, die durch einen Punkt des Eingaberaumes aktiviert wurden.
- typischerweise: Gewichtsvektor **w** bezeichnet die Gesamtheit aller Gewichte (sämtliche Gitter)

Cerebellar Model Articulation Controller (CMAC, 3)



Jede Zelle z_i wird repräsentiert durch ein binäres Feature ϕ_i:

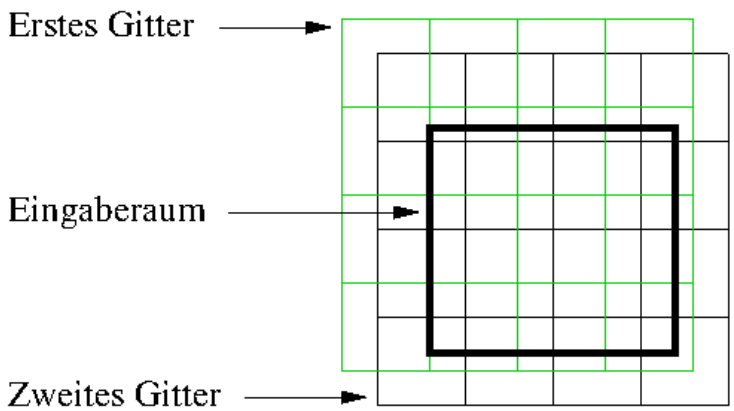
$$\mathbf{x} \in z_i \Leftrightarrow \phi_i(\mathbf{x}) = 1$$
$$\mathbf{x} \notin z_i \Leftrightarrow \phi_i(\mathbf{x}) = 0$$

$\Rightarrow$ Feature ϕ_i zeigt an, ob Eingabe $\mathbf{x}$ in Zelle z_i fällt

Darauf aufbauend: Lineares Modell

$$y(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{m} \phi_i(\mathbf{x})\mathbf{w}_i$$

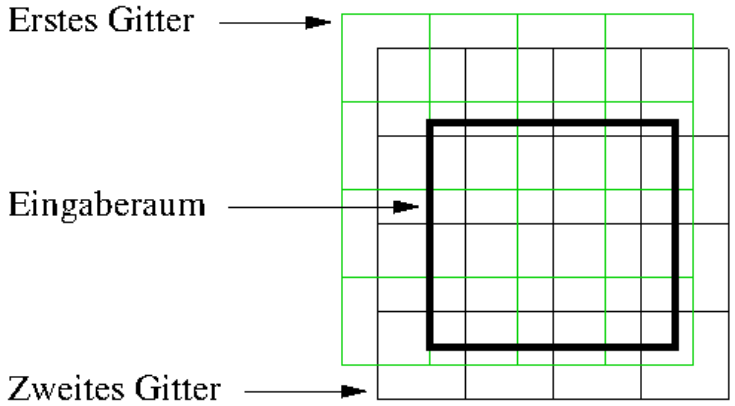
Cerebellar Model Articulation Controller (CMAC, 4)



Wie entsteht Generalisierung?

- Wert für jeden Punkt des Eingaberaums wird in verteilter Art und Weise gespeichert
- Wert für jeden Punkt des Eingaberaums wird durch mehrere Speicherzellen beeinflusst

Cerebellar Model Articulation Controller (CMAC, 5)



Lernen der Werte der Speicherzellen:

Für ein einzelnes Trainingsdatum $(\mathbf{x}, \mathbf{t})$ mit $\mathbf{x} = (x_1, \ldots, x_n)$ erfolgt eine Aktualisierung für sämtliche Gewichte (Einträge in den Gitterzellen) wie folgt:

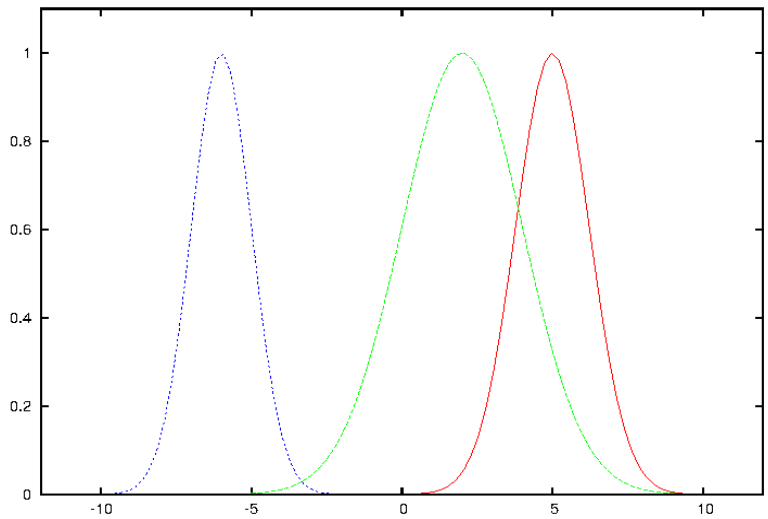
$$w_i' = \begin{cases} w_i + \frac{\alpha}{m}(\mathbf{t} - y(\mathbf{x}, \mathbf{w})) & \text{if } \phi_i(\mathbf{x}) = 1 \\ w_i & \text{if } \phi_i(\mathbf{x}) = 0 \end{cases}$$

wobei α die Lernrate repräsentiert und m für die Anzahl aktivierter Zellen (Anzahl der Gitter, $\sum_i \phi_i(\mathbf{x})$) steht.

Radiale Basisfunktionen

RBF-Netzwerke: Wahl von speziellen Basisfunktionen:

Gaussfunktion



$$\phi_i(\mathbf{x}) = \exp\left(\frac{||\mathbf{x}-\mu_i||^2}{2\sigma_i^2}\right)$$

Erwartungswert (Mittel) : μ_i

Varianz (Streuung) : σ_i

$\Rightarrow$ Basisfunktionen repräsentieren Bereiche (Cluster) im Eingaberaum

$\Rightarrow$ Überlegung: Überlappung der Verteilungen bewirkt gute Generalisierung

$\Rightarrow$ Lineares Modell: $y(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{m} \phi_i(\mathbf{x})\mathbf{w}_i$

Repräsentation der Wertfunktion im Reinforcement Learning

Überblick

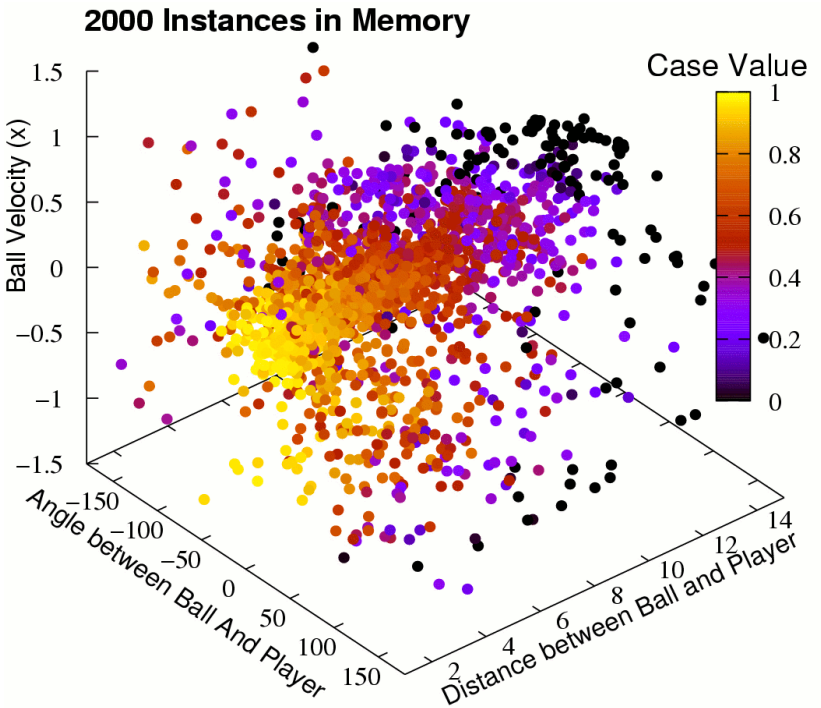
1. Motivation
2. Diskretisierung
3. Regression
4. Bäume
5. Lineare Modelle
6. Basisfunktionen
7. Instanzen- und fallbasierte Methoden
8. Neuronale Netze

Instanzen- und fallbasierte Methoden

Kernideen:

- Abspeichern von Datenpunkten über den gesamten Zustandsraum verteilt
- Suche nach nächsten Nachbarn

Beispiel einer Fallbasis



⇒ Thema wird in der Vorlesung “Fortgeschrittene Aspekte adaptiver Wissenssysteme” vertieft

⇒ Schwerpunktgebiet “Fallbasiertes Schließen” (Case-Based Reasoning, CBR)

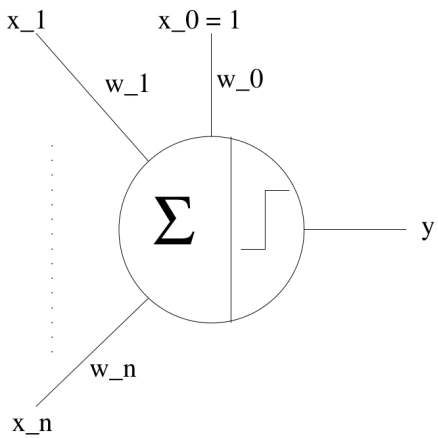
Repräsentation der Wertfunktion im Reinforcement Learning

Überblick

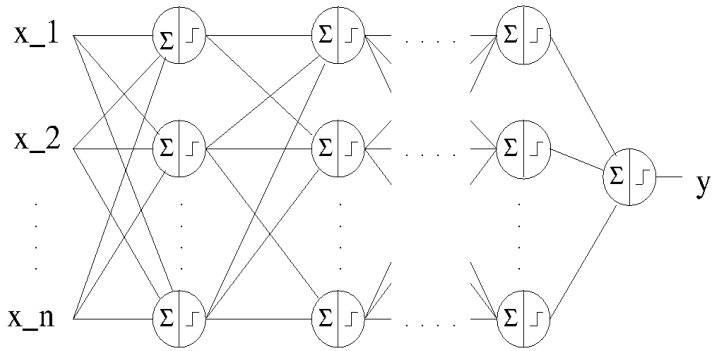
1. Motivation
2. Diskretisierung
3. Regression
4. Bäume
5. Lineare Modelle
6. Basisfunktionen
7. Instanzen- und fallbasierte Methoden
8. Neuronale Netze

Neuronale Netze: Das Multi-layer Perzeptron (MLP)

Einfaches Perzeptron:



Idee: Hintereinanderschaltung vieler Neuronen



Überwachtes Lernen im MLP

Aufgabenstellung:

- gegeben: Trainingsmenge D aus Musterpaaren $(\mathbf{x}^i, \mathbf{t}^i)$
- gesucht: Geeignete Gewichte w_{ij}

Überwachtes Lernen im MLP

Aufgabenstellung:

- gegeben: Trainingsmenge D aus Musterpaaren $(\mathbf{x}^i, \mathbf{t}^i)$
- gesucht: Geeignete Gewichte w_{ij}

Lösung:

Minimierung einer Fehlerfunktion E durch Gradientenabstieg:

- Gesamtfehlerfunktion

$$E(\mathbf{w}) := \frac{1}{2} \sum_{p=1}^{P} (y(\mathbf{x}^p, \mathbf{w}) - t^p)^2$$

- Gewichtsanpassung

$$w_{ij}^{neu} := w_{ij}^{alt} - \alpha \frac{\partial E(\mathbf{w})}{\partial w_{ij}}$$

- Berechnung $\frac{\partial E}{\partial w_{ij}}$: Backpropagation $\Rightarrow$ VL ML

Aufbau eines Neurons

Bezeichnungen

- Neuron, Unit i
- k ankommende Gewichte von Neuron j zu Neuron i: $w_{i1}, \ldots, w_{ij}, \ldots, w_{ik}$

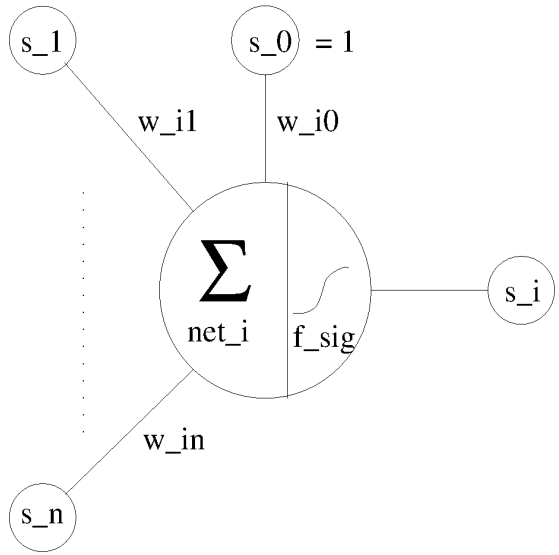
Aufbau eines Neurons

Bezeichnungen

- Neuron, Unit i
- k ankommende Gewichte von Neuron j zu Neuron i: $w_{i1}, \ldots, w_{ij}, \ldots, w_{ik}$
- Netzeingabe (interne Aktivierung):

$$net_i := \sum_{j=0}^{n} w_{ij}\, s_j$$

Modell:



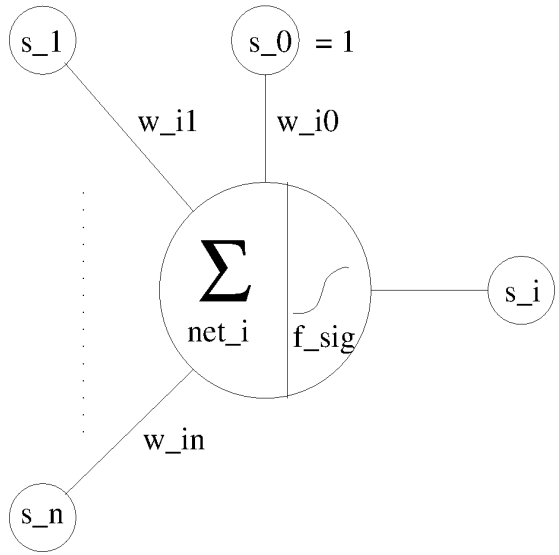
Aufbau eines Neurons

Bezeichnungen

- Neuron, Unit i
- k ankommende Gewichte von Neuron j zu Neuron i: $w_{i1}, \ldots, w_{ij}, \ldots, w_{ik}$
- Netzeingabe (interne Aktivierung):

$$net_i := \sum_{j=0}^{n} w_{ij}\, s_j$$

Modell:

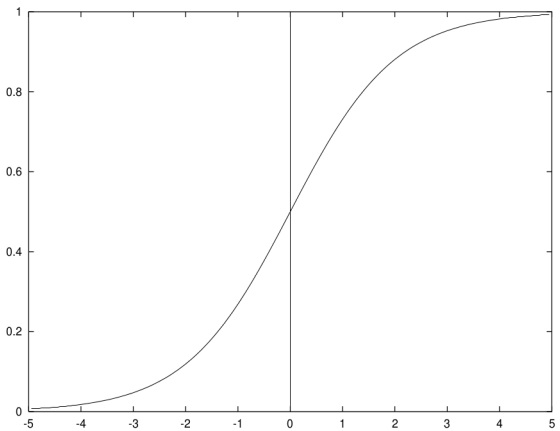


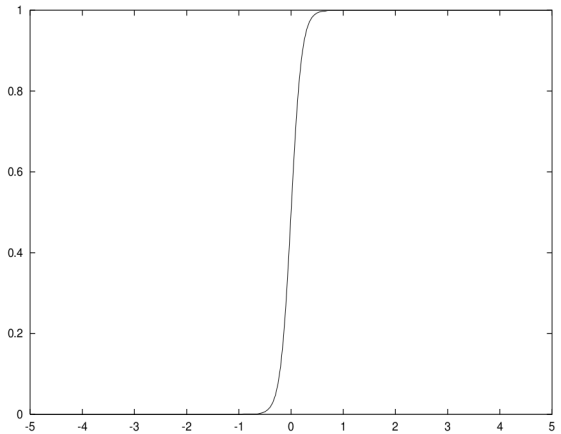
- Aktivierung bzw. Ausgabewert von Neuron i $s_i := f_{sig}(net_i)$ mit f_{sig} : Aktivierungsfunktion

Die Aktivierungsfunktion (1)

Motivation: Die Schwellwertfunktion f_{step} wird angenähert durch eine differenzierbare, monton wachsende Funktion. Hier haben sich die sogenannten Sigmoidfunktionen durchgesetzt.

$$f_{sig}(z) = \frac{1}{1 + e^{-a\,z}}$$





Die Aktivierungsfunktion (2)

Eigenschaften

- differenzierbar, streng monoton wachsend, Wertebereich zwischen 0 und 1 (manchmal auch durch einfache Transformation zwischen -1 und 1)
- für kleines *a* einen fast linearen mittleren Bereich

Die Aktivierungsfunktion (2)

Eigenschaften

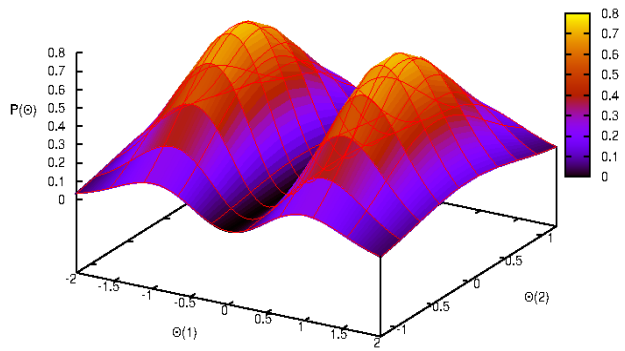
- differenzierbar, streng monoton wachsend, Wertebereich zwischen 0 und 1 (manchmal auch durch einfache Transformation zwischen -1 und 1)
- für kleines *a* einen fast linearen mittleren Bereich
- für grosses *a* nähert sich f_{sig} einer Schwellwertfunktion an
- In der Praxis wird der Parameter *a* nicht explizit verwendet; man kann zeigen, dass dieser sich durch entsprechende Wahl des Gewichtsvektors (Parametervektors) **w** ausdrücken lässt
- *tanh* und *ReLU* als weitere populäre Aktivierungsfunktionen

Lernen im MLP (1)

Probleme / Herausforderungen:

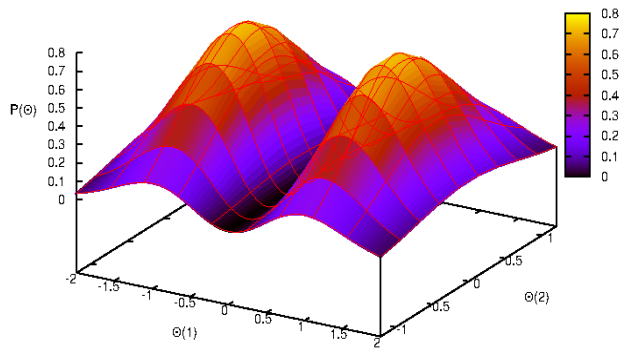
- Es gibt viele Parameter / Gewichte, um die Gesamtfunktion zu ändern, d.h. viele Freiheitsgrade.
- Wo muss wieviel gedreht werden, um *alle* Muster gleich gut zu lernen?
  - Leider: Es existiert keine eindeutige analytische Lösung wie im linearen Fall.

Lösung: Funktionsminimierung durch Gradientenabstieg



Lernen im MLP (2)

Lösung: Funktionsminimierung durch Gradientenabstieg



$$\begin{cases} \text{wenn } E'(x) > 0 & , \quad \text{verkleinere } x \\ \text{wenn } E'(x) < 0 & , \quad \text{vergrössere } x \\ \text{wenn } E'(x) = 0 & , \quad \text{lokales Minimum gefunden} \end{cases}$$

oder

$$x_{neu} := x_{alt} - \alpha\, E'(x_{alt})$$

mit $\alpha > 0$ 'Lernrate'

Lernen der Netzgewichte

Gesamtfehlerfunktion

$$E(\mathbf{w}) := \frac{1}{2}\sum_{p=1}^{P}(y(\mathbf{x}^p, \mathbf{w}) - t^p)^2$$

Gewichtsanpassung

$$w_{ij}^{neu} := w_{ij}^{alt} - \alpha \frac{\partial E(\mathbf{w})}{\partial w_{ij}}$$

vektorielle Schreibweise

$$\mathbf{w}^{neu} := \mathbf{w}^{alt} - \alpha \nabla E(\mathbf{w})$$

Lernen der Netzgewichte

mit

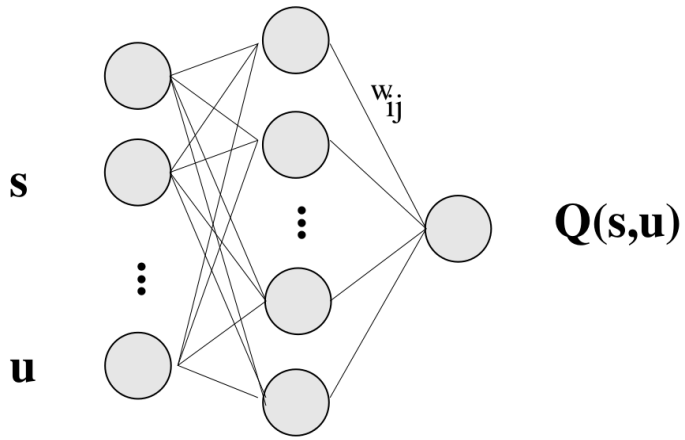
$$\nabla E(\mathbf{w}) = (\frac{\partial E(\mathbf{w})}{\partial w_1}, \frac{\partial E(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_n})^T$$

Gradient von $E(.)$ an der Stelle w_{alt}

Vorteil: Berechnung des Gradienten ist möglich: Backpropagation

$\Rightarrow$ Lernen im MLP = Gradientenabstieg

MLPs zur Repräsentation der Q-Funktion



Parameter des MLPs

- Codierung der Eingangsvariablen (Zustand)
  - Merkmale (Features)
- Skalierung der Eingangsvariablen
- Anzahl der verborgenen Neuronen
- Lernrate(n)

Zusammenfassung

- Funktionsapproximatoren zur Repräsentation von Pfadkosten bei sehr grossen/ in kontinuierlichen Zustandsräumen
- Idee: Beschreibung einer möglicherweise sehr großen Menge möglicher Eingaben durch eine begrenzte Anzahl von Parametern einer Funktion
- Diskretisierungen, Regressionsbäume
- Lineare Modelle mit Erweiterung durch Basisfunktionen
- instanzenbasierte (fallbasierte) Techniken
- neuronale Netze, Typ Multi-Layer-Perzeptron als nichtlineare Funktionsapproximatoren: Gradientenabstiegsverfahren zur Einstellung der Gewichte
- V-Funktion: Eingabe: Zustand
- Q-Funktion: Eingabe: Zustand + Aktion
- Wichtig: Codierung + Merkmale (Features)