Bandit-Based Active Learner

Seminar: Maschinelles Lernen
Referent: Markus Schelmbauer
WS 2014/15

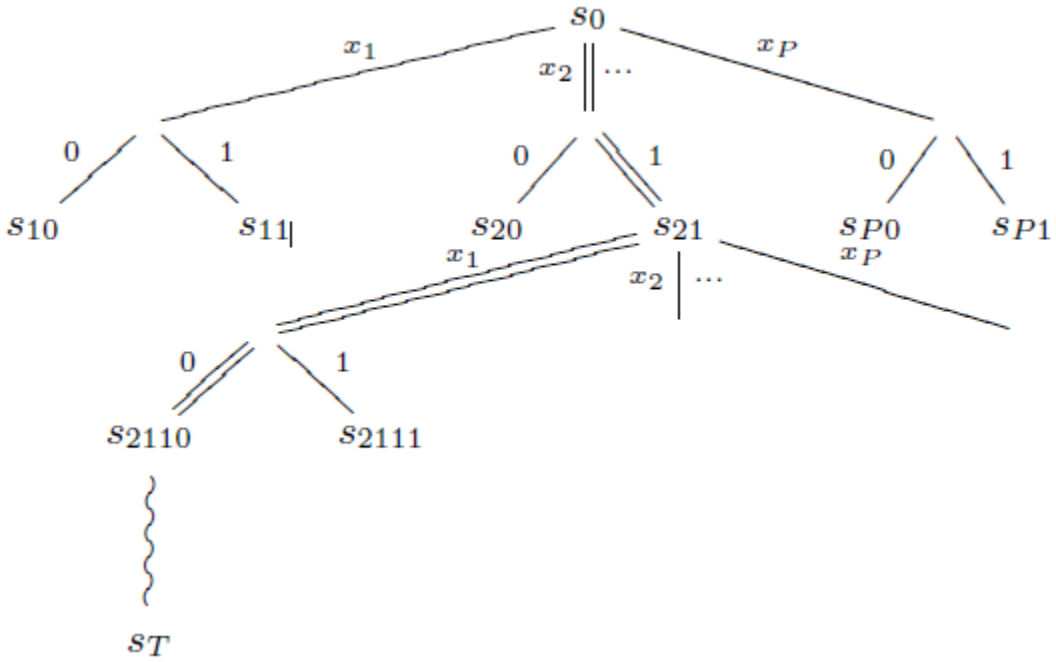


Abb.1

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Einleitung

2 Active Learning

3 BAAL

4 Billiard Algorithmus

5 Progressive Widening / QbC

6 Performance and Scalability

7 Zusammenfassung

1 Einleitung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bandit-Based Active Learner (BAAL)
- Motivation
  - bei großer Anzahl an Armen: pure exploration basiert auf UCB

Idee:

- zwei Komponenten: Exploitation vs. Exploration

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Einleitung

2 Active Learning

3 BAAL

4 Billiard Algorithmus

5 Progressive Widening / QbC

6 Performance and Scalability

7 Zusammenfassung

2 Active Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

Active Learning bedeutet z.B. Lernen unter aktiver Teilnahme des Users

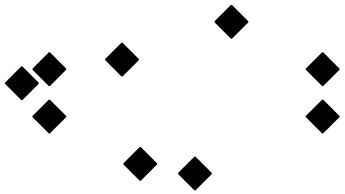
- Active Learning (AL), unter begrenzten Ressourcen, wird als “finite horizon Reinforcement Learning problem” angesehen.

- AL wird hier als Einspielerspiel betrachtet.

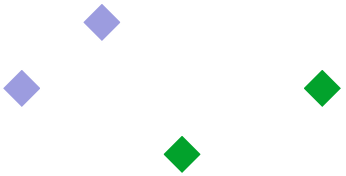
2 Active Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

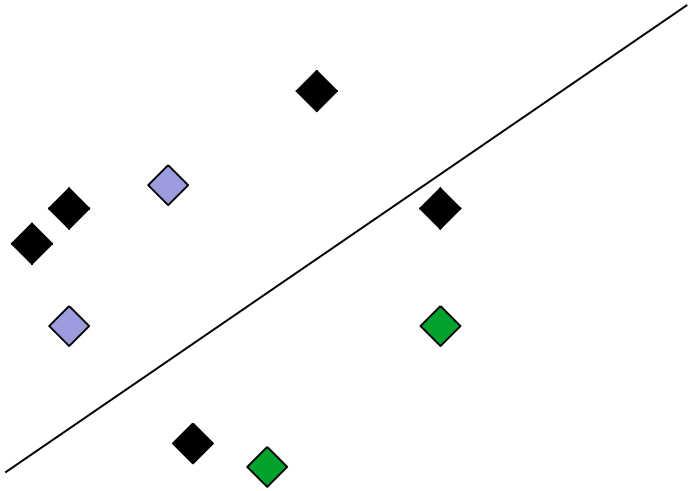
Ungelablte Daten



Gelabelte Daten



Label?



2 Active Learning

AL formalisiert als Markov Decision Process (MDP)

- Klassische MDP Zustände:
  - states, actions, reward, policy and transition functions
  - *state space* S enthält alle möglichen Trainingssets s_t
  - *action* entspricht der Auswahl einer neuen Instanz die gelabelt wird
  - *Set of actions* festgelegt durch A, das mit dem Instanz Space X übereinstimmt oder eine Teilmenge davon ist
  - mit dem state s_t verbundene *reward* function, entspricht dem Generalisierungsfehler der Hypothese $A(s_t)$ mit s_t erlernt durch einen Lerner A

2 Active Learning

- Die *transition* function $p : S \times A \times S \rightarrow \mathbb{R}_+$ definiert die Wahrscheinlichkeit der Ankunft im Zustand s_{t+1} durch Auswählen einer action x in state s_t.
- Im AL Kontext ist MDP *policy* ein sampler S, mapping a state s_t auf eine action. Dort heißt eine neue Instanz x_{t+1}

Nach dieser Definition kann AL als ein Einspielerspiel gesehen werden. Der aktive Lerner spielt gegen eine unbekannte Zielhypothese h, die zum Version Space $H(s_0)$ des initialisierten Trainingssets s_0 gehört.

2 Active Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

Nach jedem Zug (das Aussuchen einer Instanz x) entscheidet das oracle h das lable $y = h(x)$.

Am Ende des Spiels, nach dem T Samples ausgewählt worden sind, definiert von Trainingsset s_T, ist der reward der Generalisierungsfehler der Hypothese $A(s_T)$ gelernt von s_T.

2 Active Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

Es ist möglich, einen AL Spieler zu trainieren und hierzu eine gute AL policy zu entwickeln:

- Vorheriges Spiel und Spielen gegen ein Ersatz oracle (eine Hypothese h in VS ausgewählt) nachahmen

Der reward wird wie in einem richtigen Spiel berechnet. Es ist der Generalisierungsfehler der erlernten Hypothese gegenüber dem Ersatz oracle. Dieser soll minimiert werden.

$$S_T^* = \arg\min_S \mathbb{E}_{h\sim\mathcal{H}}\, \mathbf{Err}(A(S_T(h)), h).$$

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Einleitung

2 Active Learning

3 BAAL

4 Billiard Algorithmus

5 Progressive Widening / QbC

6 Performance and Scalability

7 Zusammenfassung

3 BAAL – Bandit-Based Active Learner

TECHNISCHE UNIVERSITÄT DARMSTADT

Zwei Hauptbestandteile:

1. tree-structured multi-armed bandit, das UCT erweitert zu einem Einspielerspiel im AL

2. Ein fairer und sparsamer billiard-based Algorithmus, um die Instanz und Hypothesenräume zu sampeln.

3 BAAL – Bandit-Based Active Learner

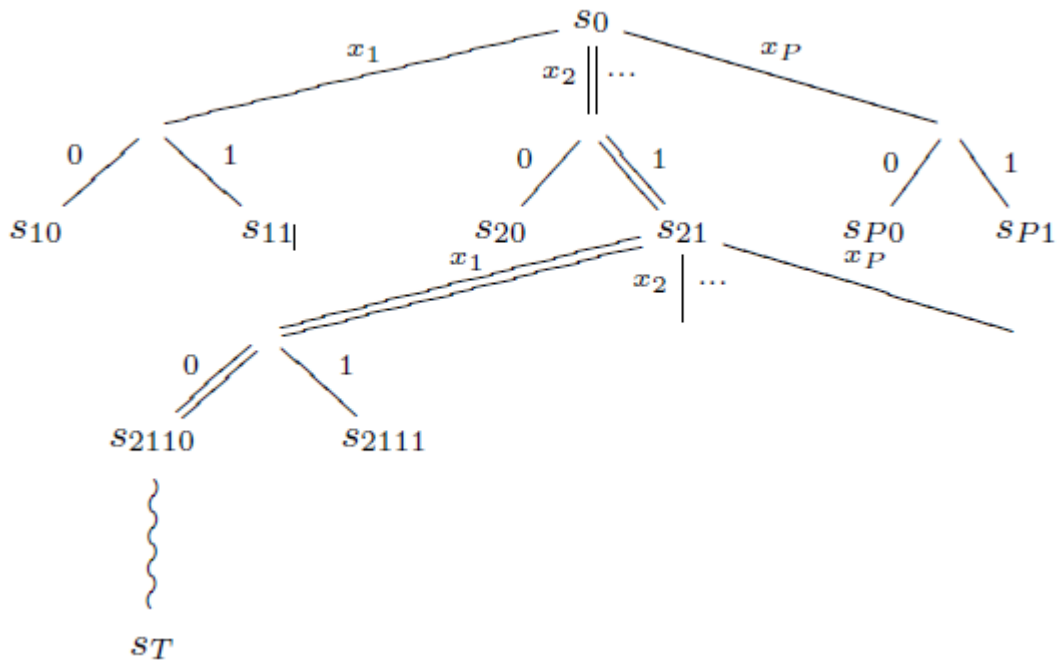
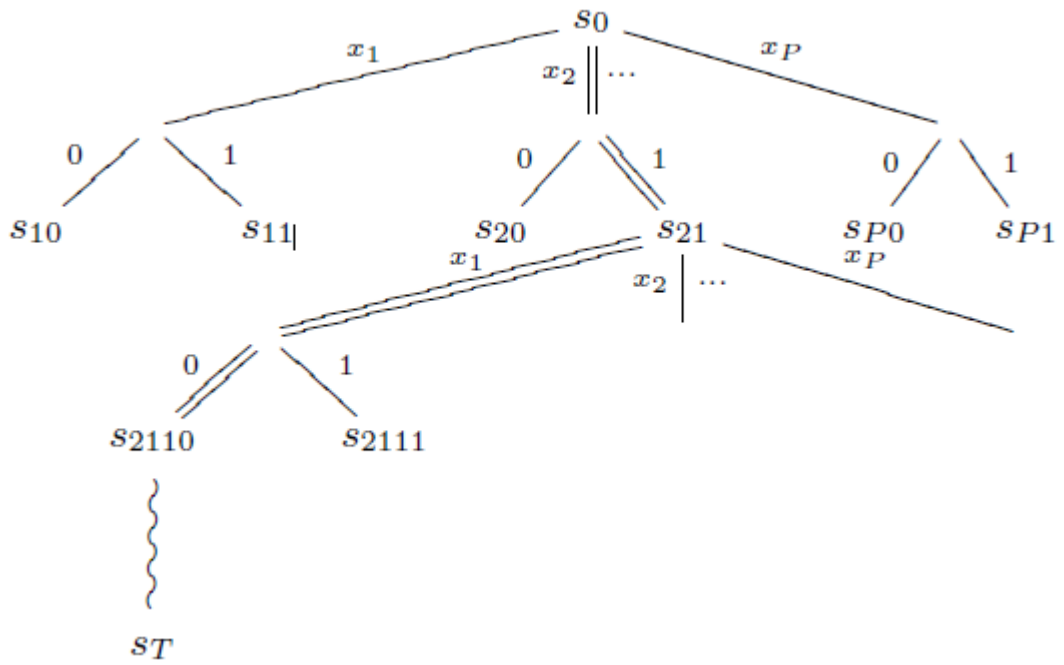


Abb.1

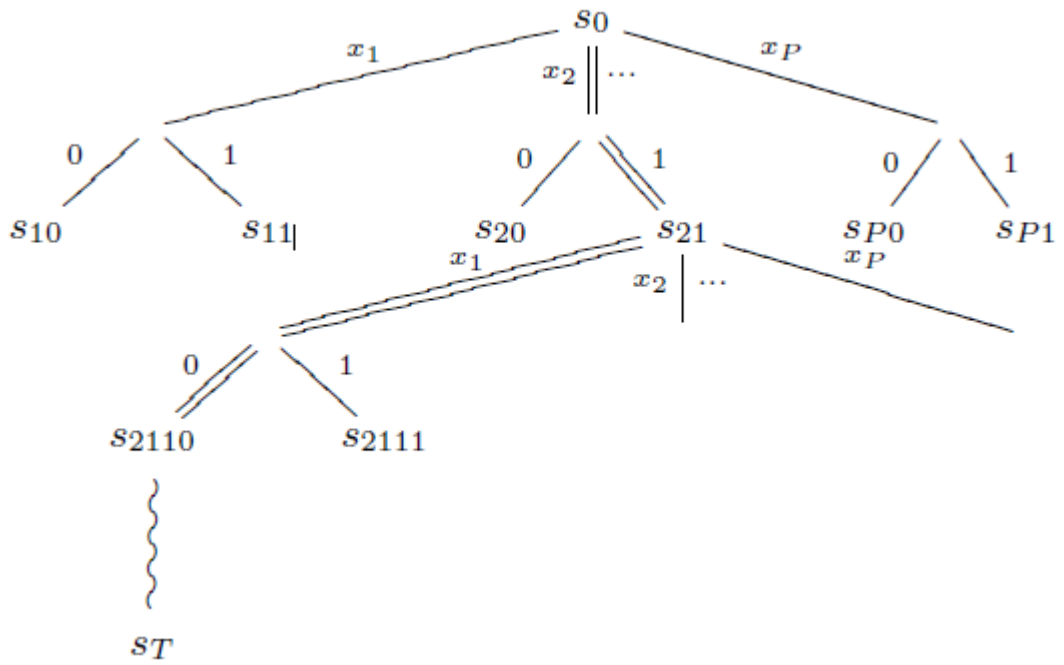
3 BAAL – Bandit-Based Active Learner



Knoten, der einem Trainingsset s_t entspricht → Zustandsknoten.

Abb.1

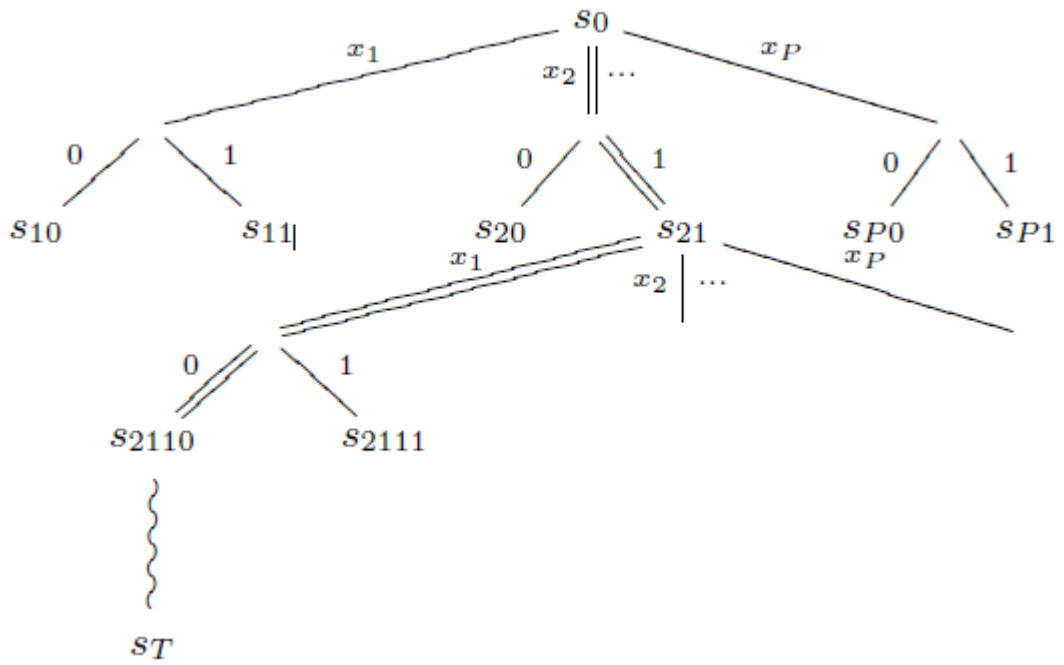
3 BAAL – Bandit-Based Active Learner



Nachfolgende Knoten sind Entscheidungsknoten, wenn:
sie aus Trainingsset s_t und der gewählten Instanz x_{t+1} bestehen

Abb.1

3 BAAL – Bandit-Based Active Learner



Jeder Entscheidungsknoten hat zwei Kindknoten (mögliche Label der Instanz x_{t+1}).

- durch tree-walk ausgewählter Knoten entspricht $\mathrm{h}(x_{t+1})$ und führt zum nächsten Zustandsknoten $s_{t+1} = s_t \cup (x_{t+1}, h(x_{t+1}))$.

Abb.1

3 BAAL - Definitionen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hypothesen H
- P_H Hypothesen Verteilung
- Versionspace H(s0)
- Label = Hypothese angewandt auf Instanz
- Set s = {(x, h(x))}
- Endzeit T
- N : Performancefactor, Anzahl der Tree-Walks

3 BAAL

Algorithm 1 The ***BAAL*** algorithm:
Input: measure P_H on hypothesis space $\mathcal{H}$; initial training set s_0; time horizon T; number N of allowed tree-walks;
Output: an instance x to be labelled by the oracle.

BAAL(P_H, s_0, T, N)
for i=1 to N **do**
  $h =$ `DrawHypothesis`(P_H, s_0)
  Tree-Walk(s_0, T, h)
end for
Return $x = \arg\max_{x' \in \mathcal{X}} \{n(s \bigcup \{x'\})\}$

- *DrawHypothesis* wählt die Ersatzhypothese h im Anhang zu jedem tree-walk unter Benutzung des Billiard Algorithmus.

3 BAAL - Tree-Walk

TECHNISCHE UNIVERSITÄT DARMSTADT

Tree-Walk(s, t, h)
Increment $n(\mathrm{s})$
if t==0 **then**
  Compute $r = Err(\mathcal{A}(\mathrm{s}), h)$
else
  $\mathcal{X}(\mathrm{s}) =$ `ArmSet`$(\mathrm{s}, n(\mathrm{s}))$
  Select $x^* = UCB(\mathrm{s}, \mathcal{X}(\mathrm{s})))$
  $r =$ Tree-Walk$(\mathrm{s} \bigcup \{(x^*, h(x^*))\}, t-1, h)$
end if
$r(\mathrm{s}) \leftarrow (1 - \frac{1}{n(\mathrm{s})}) r(\mathrm{s}) + \frac{1}{n(\mathrm{s})}\ r$
Return r

- *ArmSet* erweitert UCT, um mit unendlich (großen) Sets von action (dem kontinuierlichen Instanz raum) umzugehen.
- Unter Benutzung des Progressive Widening

3 BAAL - Tree-Walk

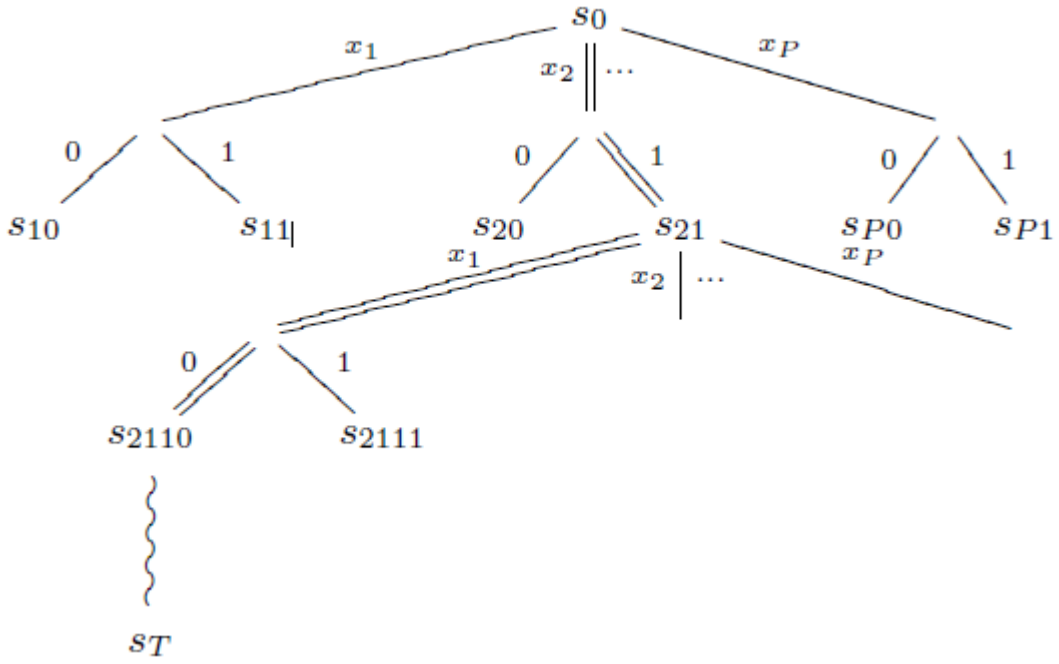


Abb.1

3 BAAL - Tree-Walk

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eine Ersatz Hypothese wird einheitlich aus Version Space H(s_0) des initialisierten Trainingssets s_0 gezogen
- In jedem Zustandsknoten (d.h. in einem Trainingsset s_t), nutzt BAAL die UCB Kriterien, um einen Entscheidungsknoten auszusuchen, d.h. eine Instanz x_{t+1} zu labeln. Das zugehörige label wird auf $\mathrm{h}(x_{t+1})$ gesetzt, während h die Ersatz Hypothese ist. Der nächste Zustandsknoten: $s_{t+1} = s_t \cup (x_{t+1}, h(x_{t+1}))$.

3 BAAL - Tree-Walk

TECHNISCHE UNIVERSITÄT DARMSTADT

- Tree-walk läuft bis zum Blatt, d.h. solange ein Zustandsknoten s_{t_o} noch nicht besucht ist. An diesem Punkt, sind T – t_0 zusätzliche Instanzen gleichmäßig ausgewählt, nach h gelabelt und zum Trainingsset s_{t_0} hinzugefügt, diese bilden ein T-großes Trainingsset s_T.
- Durch das Trainingsset s_T wurde eine Hypothese $\hat{h}$ von BAAL gelernt.
- Der reward ist der Generalisierungsfehler von $\hat{h}$ in Bezug auf die Ersatz Hypothese h: dieser reward wird genutzt um den Wert von jedem relevanten Knoten zu updaten.

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Einleitung

2 Active Learning

3 BAAL

4 Billiard Algorithmus

5 Progressive Widening / QbC

6 Performance and Scalability

7 Zusammenfassung

4 Billiard Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jeder Tree-walk in BAAL wird durch eine Hypothese h indiziert.
- Die einfache Lösung ist ein sampling Algorithmus der auf rejection beruht.
- Hypothesen werden konstant in H gezogen und rejectet wenn sie nicht in den VS passen.

Alternative Algorithmen:

- Gibbs sampling oder Monte Carlo Markov Chains (MCMC) Methoden beinhalten wenige freie Parameter und skalieren möglicherweise schlecht in Bezug auf die Dimensionalität des Suchraums und der Größe der Trainingsmenge s_0.

4 Billiard Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

Problem: Rejection ist nicht tractable

- billiard sichert mit constraints ab, dass die ausgewählte Hypothese immer auf das Trainingsset passt.

Annahme: Hypothesen sind parametrisierbar mit d Parametern.

- $\mathbb{R}^d$: Hyphotesenraum
- Ω: Subraum des Hypothesenraums, der Raum in dem die Hypothesen bestimmte Constraints{g(x)} erfüllen. constraints $g_i, \ldots, g_n$: = {x ∈ $\mathbb{R}^d$ s.t. g_i(x) ≥ 0, i = 1 . . . n}
- Startpunkt z
- Richtung v

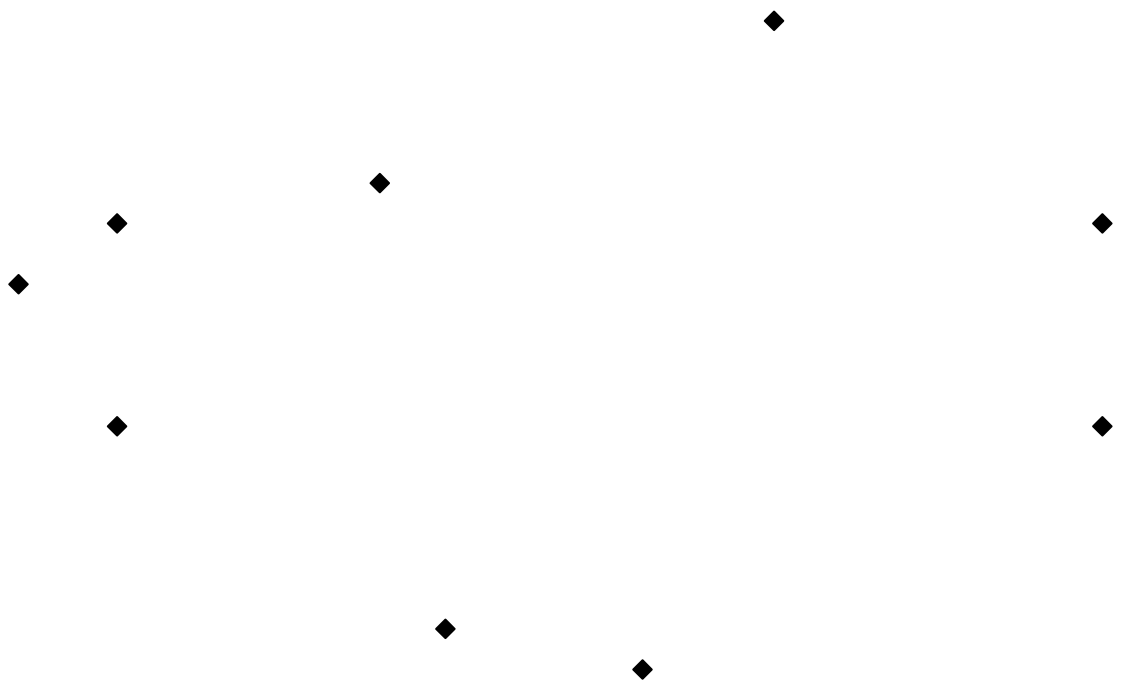
4 Billiard Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

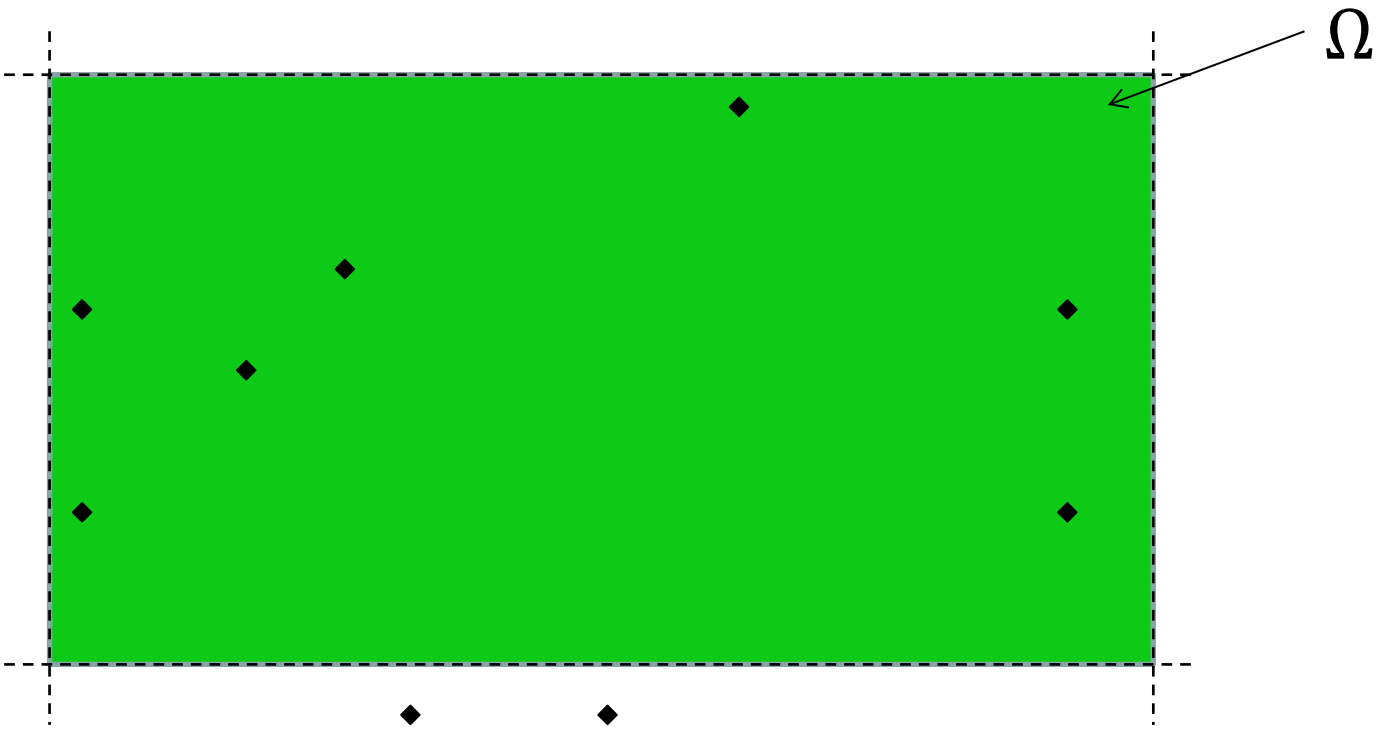
Input:

- Set von constrains g_i.
- Länge L
- Gibt einen finalen Punkt zurück.

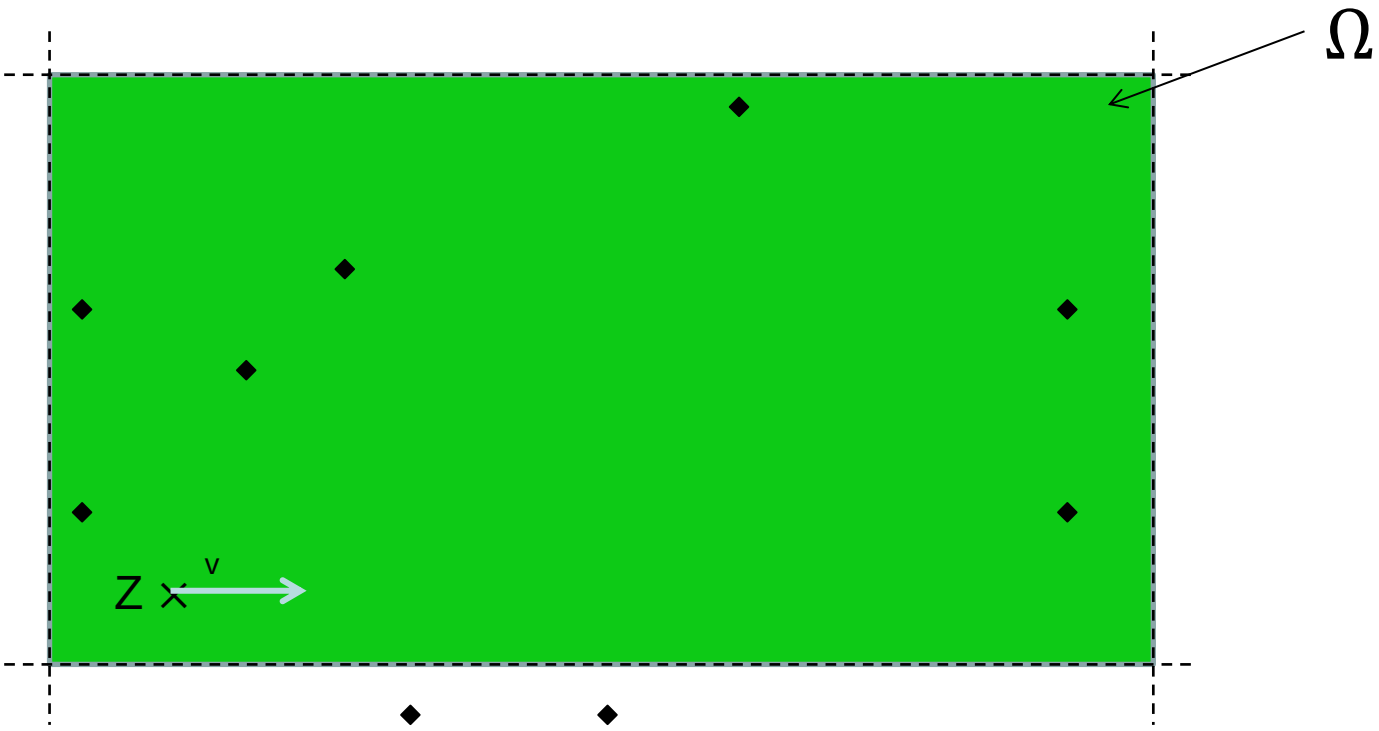
4 Billiard Algorithmus



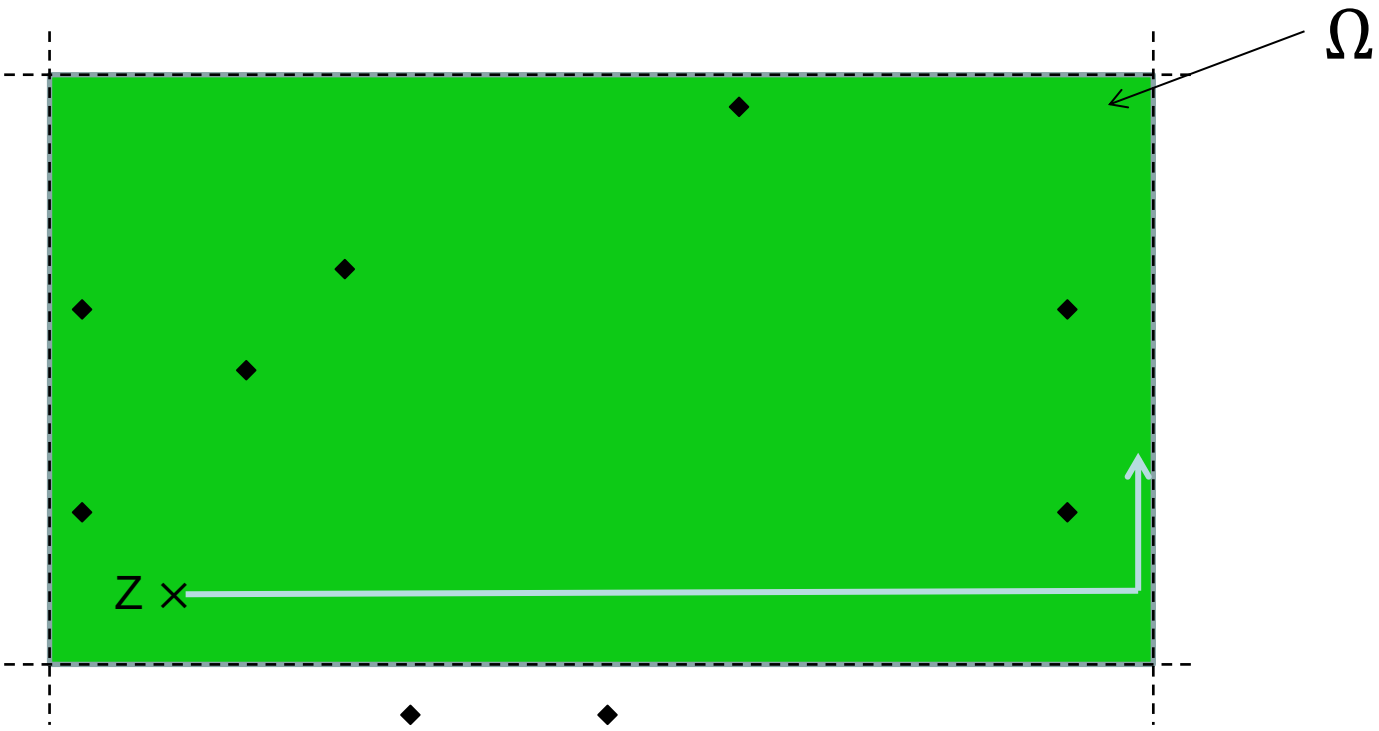
4 Billiard Algorithmus



4 Billiard Algorithmus



4 Billiard Algorithmus



RayTracing

- Reflektiert an den Grenzen (durch constraints)
- Ende nach Strahllänge L

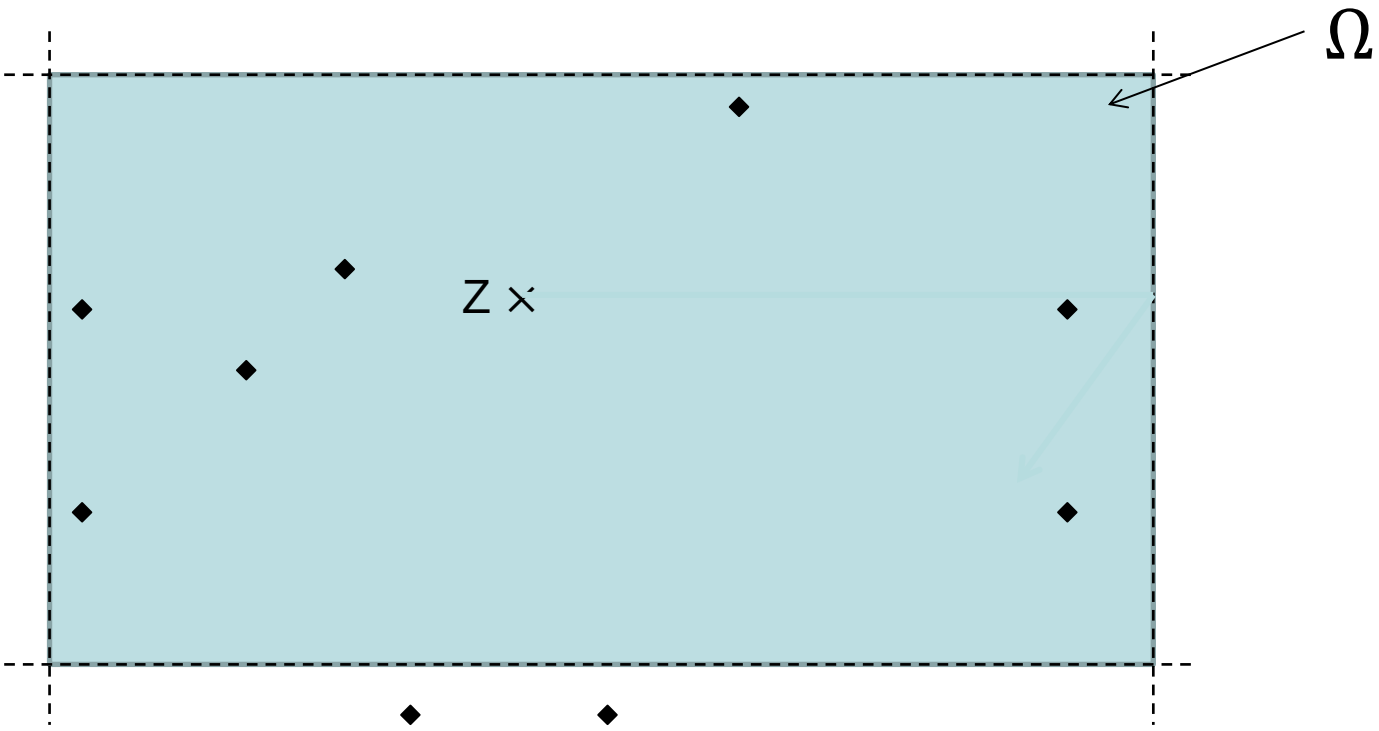
4 Billiard Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

Unter einigen Bedingungen der constraints, ist die Länge ***ergodic***.

- D.h. es füllt die ganze Domain aus, wenn L gegen Unendlich geht.

4 Billiard Algorithmus



4 Billiard Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

Billiard Algorithmus wurde erfolgreich im Maschinellen Lernen eingesetzt, z. B. um den bayes classifier in einem kernel feature space abzuschätzen(Rujan, Playing billiards in version space. Neural Computation)

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Einleitung

2 Active Learning

3 BAAL

4 Billiard Algorithmus

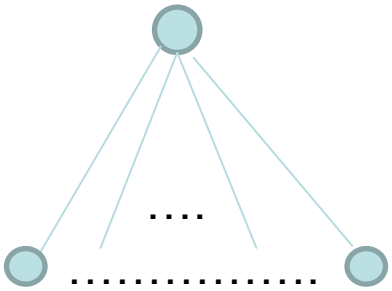
5 Progressive Widening / QbC

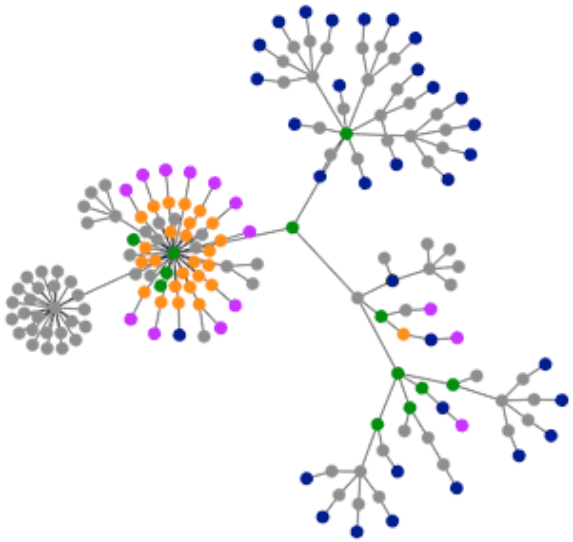
6 Performance and Scalability

7 Zusammenfassung

5 Progressive Widening

- UCB verlangt ursprünglich von jedem Arm, mindestens einmal ausgewählt zu werden.
- Wenn die Anzahl der Arme, in Bezug auf die Anzahl der Simulationen, größer ist, neigt UCB in eine reine Exploration zu entarten.





- UCT stellt die gleichen Einschränkungen unter Berücksichtigung von viele Armen

Abb.2

5 Progressive Widening

TECHNISCHE UNIVERSITÄT DARMSTADT

- Progressive Widening (PW) wurde vorgestellt (Coulom, R) um mit solchen Situationen umzugehen
- n_s ist die Anzahl, die Knoten s bisher besucht wurde
- Die Anzahl der Arme, die von s in Betracht gezogen werden können, ist limitiert auf ein Bruchteil m von n_s

Empirische und theoretisch Studien suggerieren $m = O\left(n_s^{1/4}\right)$

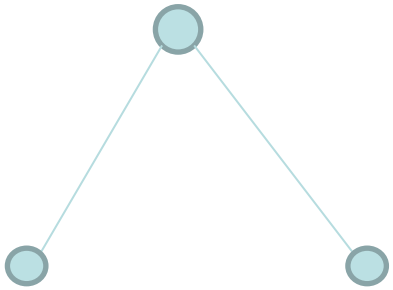
5 Progressive Widening

- Problem: pure exploration bei zu vielen Armen

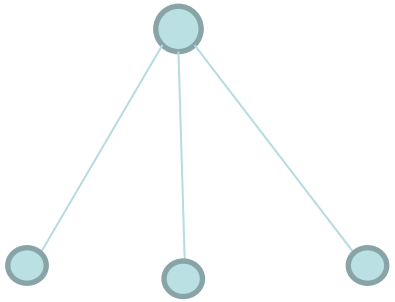
- Heuristik: $m = \lfloor n^{(1/4)} \rfloor$ ist die Anzahl der benutzbaren Arme

1,16,81,256,...: 1,2,3,4,... Arme

Ab 16 Besuchen



Ab 81 Besuchen



5 Progressive Widening

TECHNISCHE UNIVERSITÄT DARMSTADT

Mögliche Erweiterung

- Durch einen Signalton wird man aufgefordert, sich für einen Arm zu entscheiden.
- Entspricht der Arm den Kriterien, wird er hinzugefügt
- ArmSet implementiert PW in BAAL, und berücksichtigt ein endliches Set an Optionen für jeden Knoten.

5 Query-by-committee(QbC) BAAL

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verkleinern vom Version Space
- Maximale Unsicherheit MU ist ein Kriterium aus dem Query-by-committee Algorithmus.

Algorithmus:

- Zufälliges sampling von Hypothesen im Version Space
- wählt eine gegebene Instanz, aber nur wenn genügend Hypothesen nicht mit dem Label übereinstimmen.

5 Query-by-committee(QbC) BAAL

TECHNISCHE UNIVERSITÄT DARMSTADT

Der Ansatz ist wie folgt in BAAL integriert.

- Wann immer eine neue Instanz zum ArmSet hinzugefügt wird, wählt das committee die, die das disagreement maximiert.
- MU ist ein aggressives Kriterium
- Wann immer ein Kriterium noch nicht optimal ist, wird die limitierte Erforschung einen schlechten Ertrag ergeben.
- Es ist rechnerisch anspruchsvoller

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Einleitung

2 Active Learning

3 BAAL

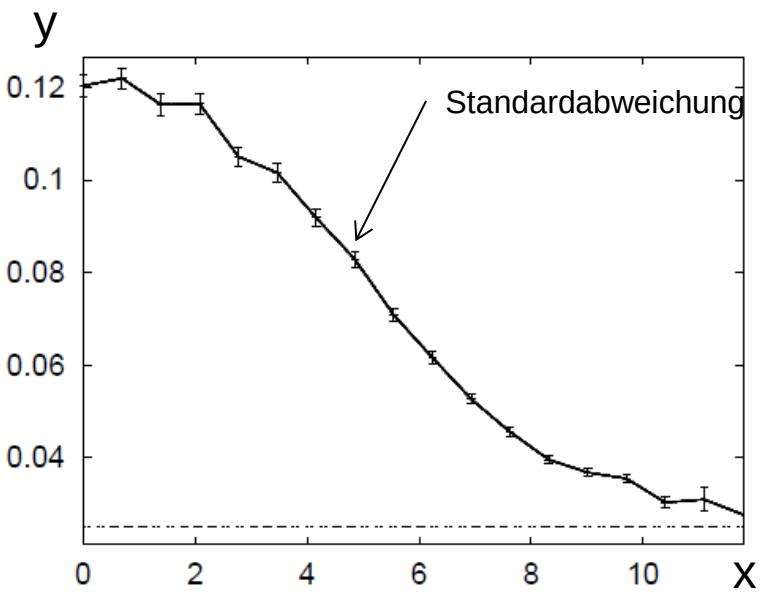
4 Billiard Algorithmus

5 Progressive Widening / QbC

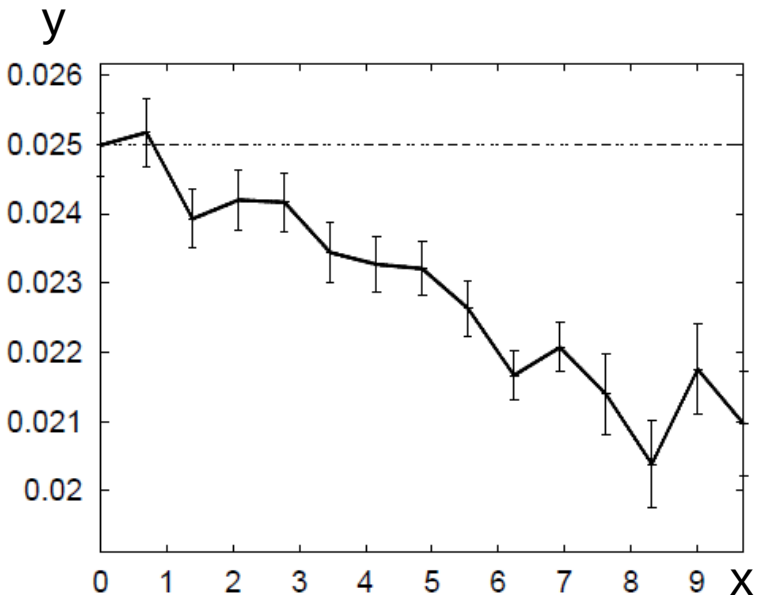
6 Performance and Scalability

7 Zusammenfassung

6 Performance and Scalability



(a) d = 4, stand-alone $BAAL$

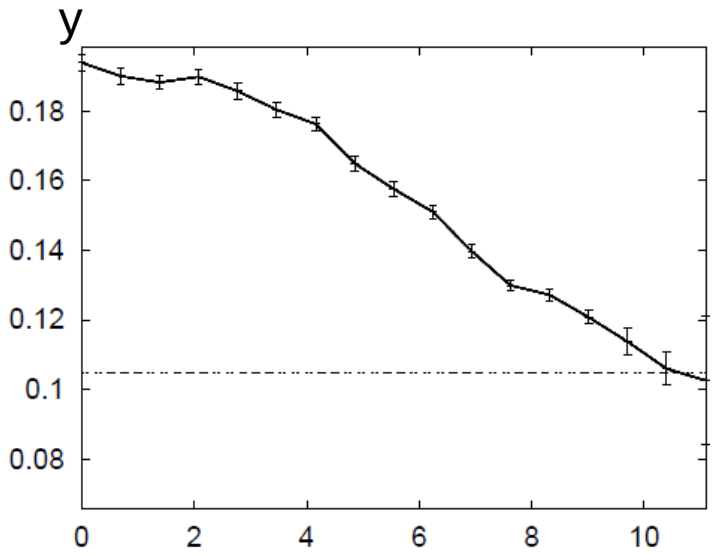


(c) d = 4, QbC-$BAAL$

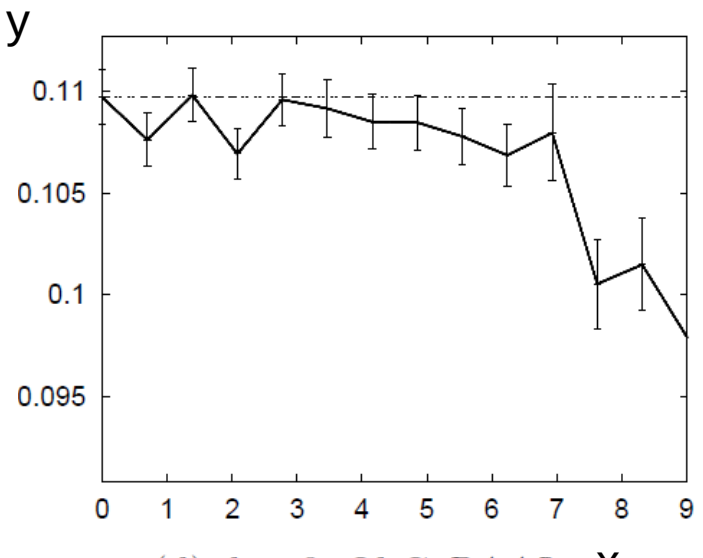
y = Generalisierungsfehler
x = Anzahl der Simulationen N

Abb.3

6 Performance and Scalability



(c) d = 8, stand-alone $BAAL$



(d) d = 8, QbC-$BAAL$

y = Generalisierungsfehler
x = Anzahl der Simulationen N

Billiard Algorithmus erlaubt eine geringere Zeit Komplexität, mehrere Größenordnungen als der rejektion Algorithmus.
Billiard braucht 17sec und rejection mehr als 3 Stunden(d = 8, T = 20,N = 16,

Abb.3

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Einleitung

2 Active Learning

3 BAAL

4 Billiard Algorithmus

5 Progressive Widening / QbC

6 Performance and Scalability

7 Zusammenfassung

7 Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beschränkte Anzahl von Abfragen. Motiviert durch Anwendung im Numerical Engineering. In diesem Bereich brauchen Durchläufe meist mehrerer Tage.
- Wurde als Reinforcement Learning Problem angegangen: Bei der Sampling Strategie ist das Ziel, den Generalisierungsfehler zu minimieren für einen endlichen Horizont.
- Eine Annäherung an die optimale sampling Strategie wird erlernt anhand eines Einspielerspiel.
- BAAL ist inspiriert durch Computer Go und aufbauend auf den Bandit-based Algorithmus, UCT und Progressive Widening Heuristik

Abbildungsverzeichnis

Abb. 1: Search Tree developed by BAAL (Fig. 1.)S.7,. Zugriff November 2014 unter *Philippe Rolet, Michele Sebag, Olivier Teytaud. Boosting Active Learning to Optimality: a Tractable Monte-Carlo, Billiard-based Algorithm. ECML, Dec 2008, Bled, Slovenia. pp.302- 317. <inria-00433866>)*

Abb. 2: Baum. Zugriff November 2014 unter http://www.netzallee.de/design/baumstruktur-nachher.png

Abb. 3: Ergebnisse (Fig. 2.)S14,. Zugriff November 2014 unter *Philippe Rolet, Michele Sebag, Olivier Teytaud. Boosting Active Learning to Optimality: a Tractable Monte-Carlo, Billiard-based Algorithm. ECML, Dec 2008, Bled, Slovenia. pp.302- 317. <inria-00433866>)*

Literaturverzeichnis

Philippe Rolet, Michele Sebag, Olivier Teytaud. Boosting Active Learning to Optimality: a Tractable Monte-Carlo, Billiard-based Algorithm. ECML, Dec 2008, Bled, Slovenia. pp.302-317. <inria-00433866>"Upper Confidence Trees and Billiards for Optimal Active Learning," in Proc. Conf. l'Apprentissage Autom., Hammamet, Tunisia, 2009

"Upper Confidence Trees and Billiards for Optimal Active Learning," in Proc. Conf. l'Apprentissage Autom., Hammamet, Tunisia, 2009.

"Optimal Robust Expensive Optimization is Tractable," in Proc. 11th Annu. Conf. Genet. Evol. Comput., Montreal, Canada, 2009, pp. 1951–1956.

Ruj ´an, P.: Playing billiards in version space. Neural Computation 9(1) (January 1997) 99–122 Ruj ´an, P., Marchand, M.: Computing the bayes kernel classifier (1999)

TECHNISCHE UNIVERSITÄT DARMSTADT

Vielen Dank
für Ihre Aufmerksamkeit!