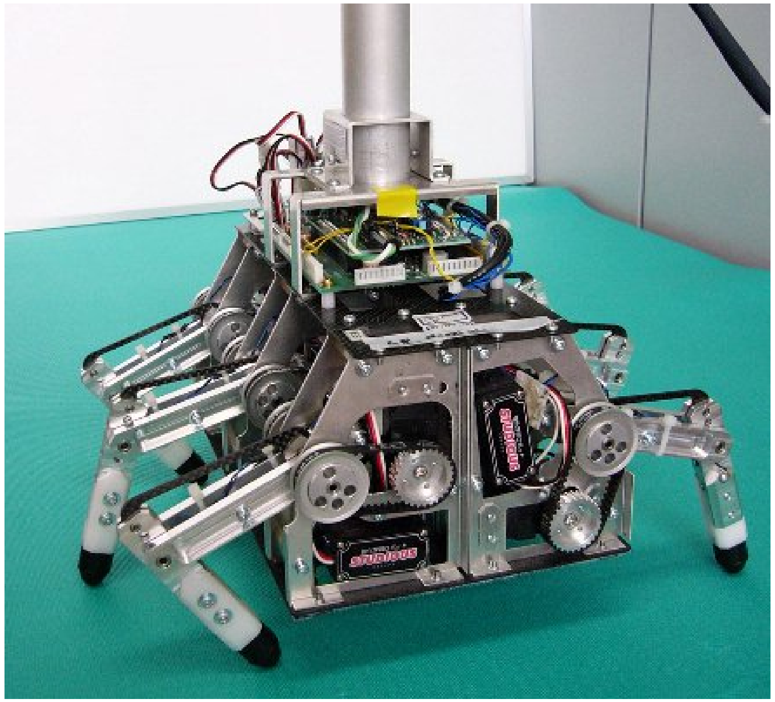
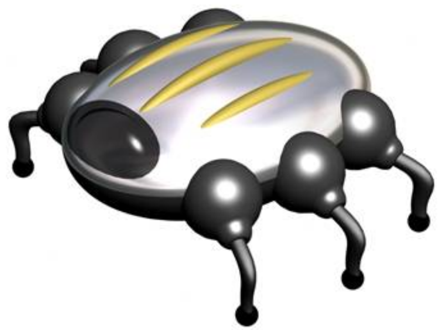
多次元状態-行動空間での強化学習

ランダム矩形タイルによる汎化方法と
Gibbsサンプリングによる行動選択方法の提案

木村　元
九州大学　大学院工学研究院海洋システム工学部門





http://sysplan.nams.kyushu-u.ac.jp/gen/index.html

ロボットの知能化

→　強化学習：試行錯誤を通じて制御規則を獲得

学習アルゴリズムに対する要請：

1）とにかく**単純**な計算処理
- ●プログラムコードが短いこと
- ●自然言語で簡潔に表現できるのが望ましい

アルゴリズム中に「微分記号」など論外！

2）**安定**した学習動作
- ●問題に依存せずに実装可能であること
- ●パラメータ設定にセンシティブではないこと

Actor-Critic　　　　Q-learning

ロボットの知能化

→　強化学習：試行錯誤を通じて制御規則を獲得

学習アルゴリズムに対する要請：

1）とにかく**単純**な計算処理
- ●プログラムコードが短いこと
- ●自然言語で簡潔に表現できるのが望ましい

アルゴリズム中に「微分記号」など論外！

2）**安定**した学習動作
- ●問題に依存せずに実装可能であること
- ●パラメータ設定にセンシティブではないこと

~~Actor-Critic~~　　**Q-learning**

本研究の目的

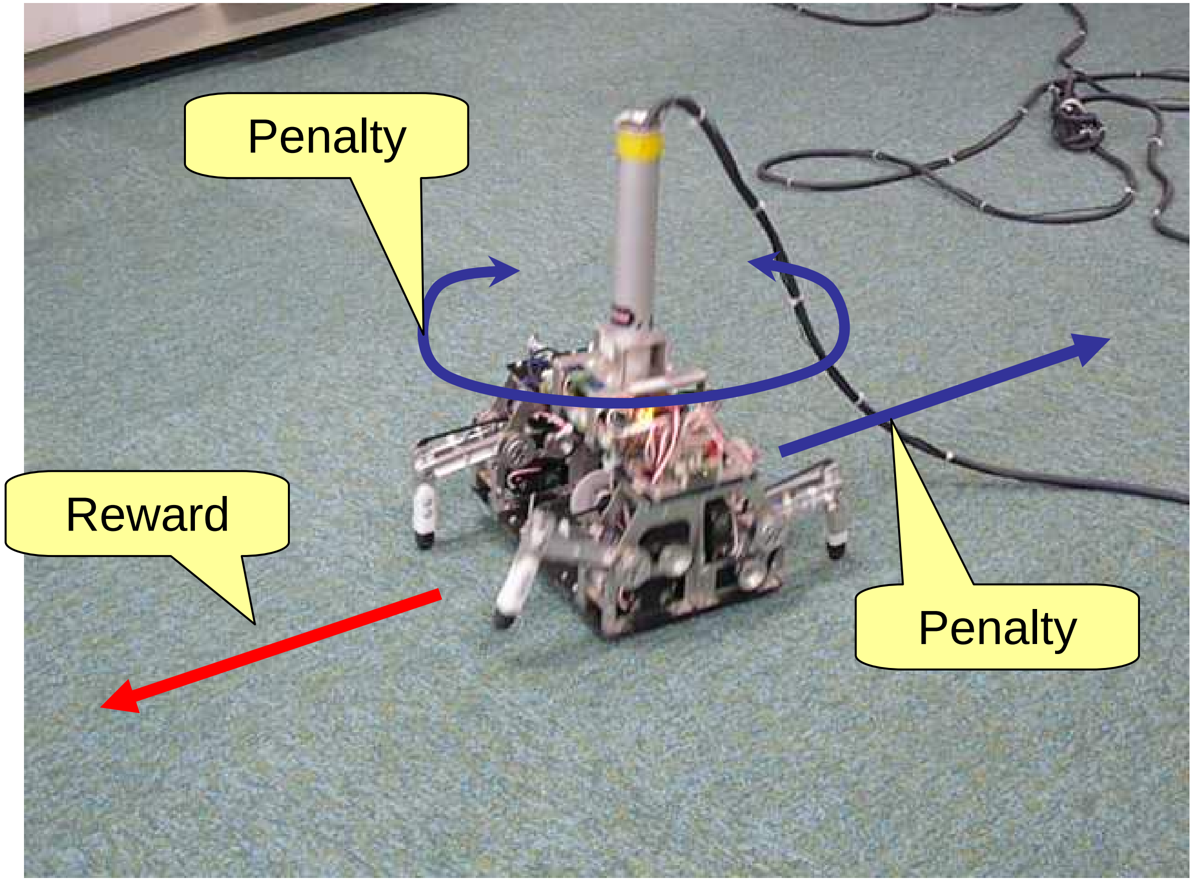
高次元連続状態-行動空間のロボット強化学習

Q-learning	連続空間の関数近似	ボルツマン行動選択

・これら基礎的な技術と知見をそのまま継承した実装
・数万ステップで学習させる（Actor-Criticの数倍程度以内）

解決すべき課題：
1）状態-行動の評価値（Q値）の表現　←膨大な空間の汎化
2）高次元連続行動空間における行動選択は？

The proposed Learning method is applied to 4-legged real robot to learn walking behavior.



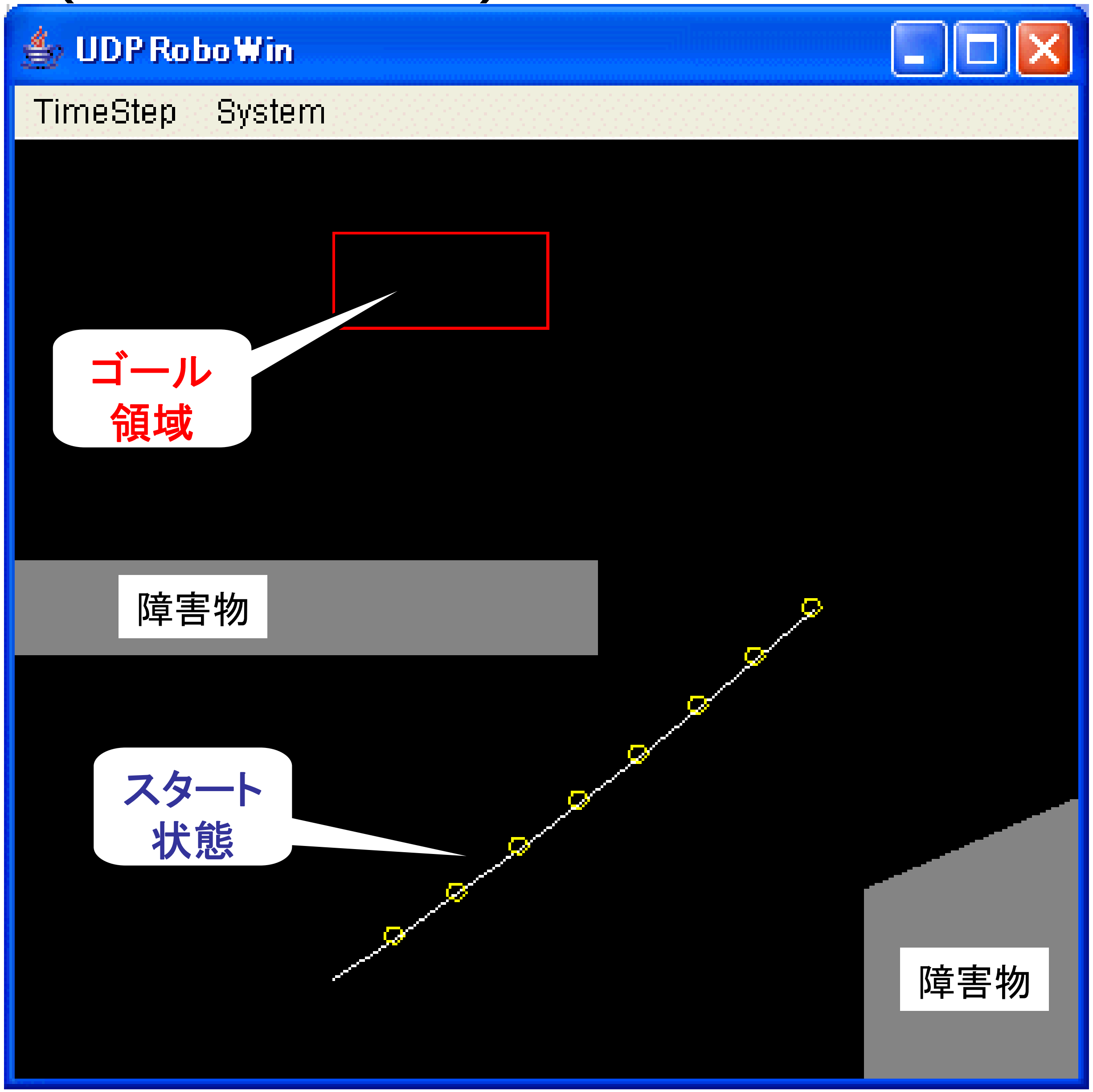
0.5 sec/step

State is the current angles of the Joints (8 dimensional).

Action is destination angles of the joints (8 dimensional)

The agent found a policy: “stopping” to avoid penalties.

対象とする学習問題：　Multi-Joint Arm (Moore 1995)



2次元平面中でスタート状態からゴール領域までアームを移動

領域中に固定障害物

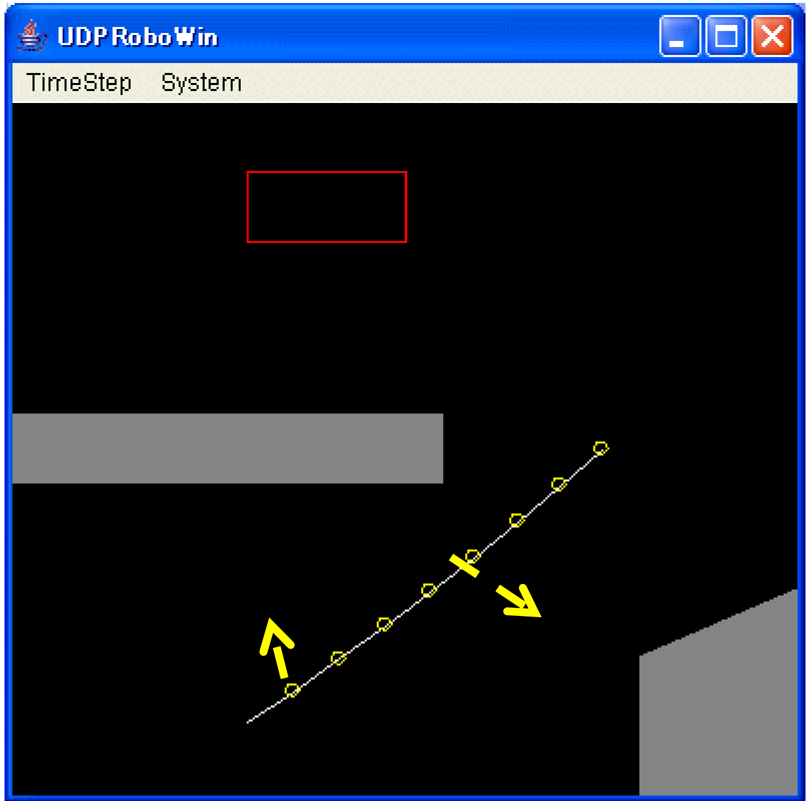
アームは冗長自由度（8関節）

状態：　関節角度θ（8次元）

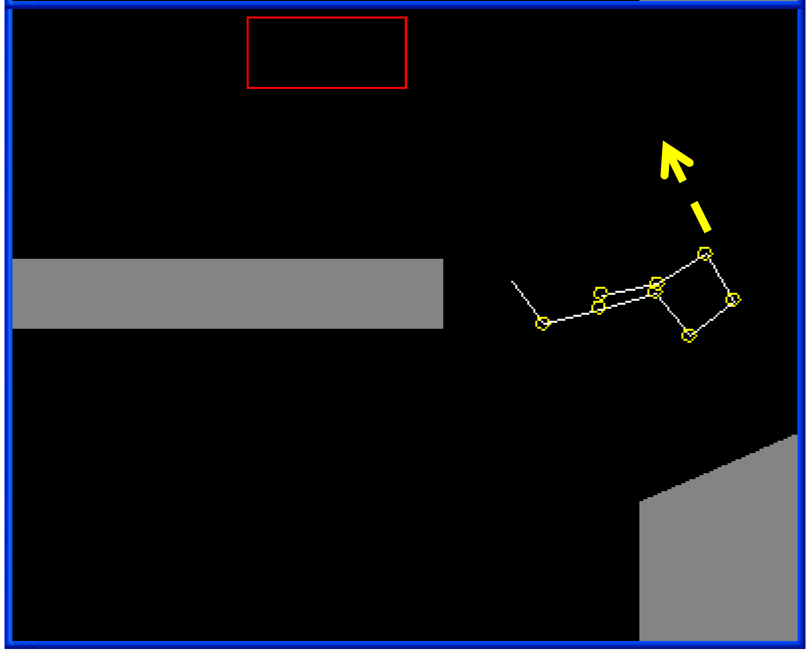
行動：　目標角度θ（8次元）

アームが目標角度へ到達するか、途中で障害物へぶつかると意思決定のイベント発生

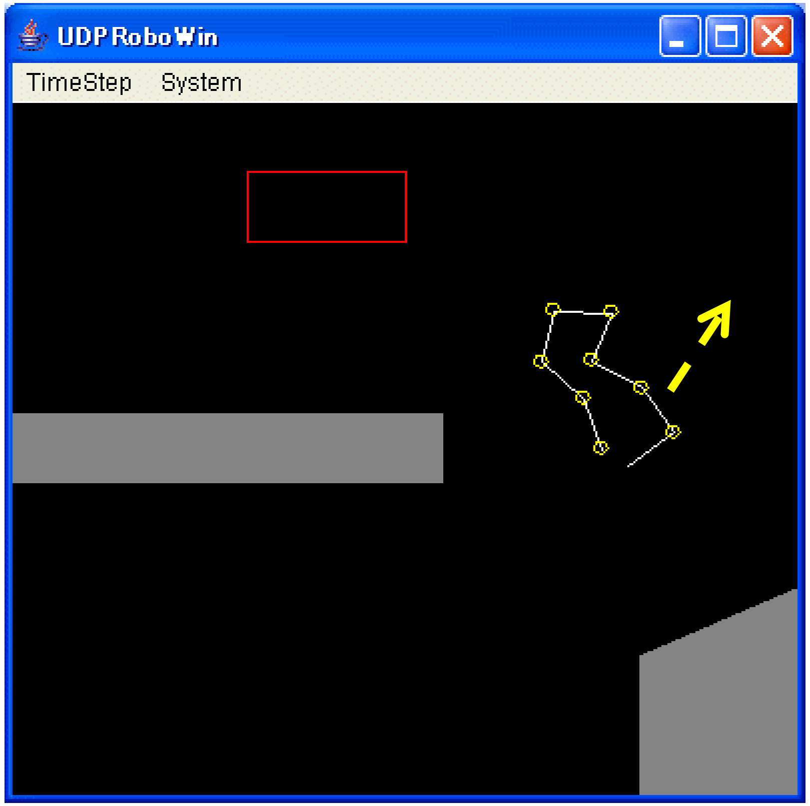
ゴールへ到達すると報酬／再び初期状態へもどる



(1)

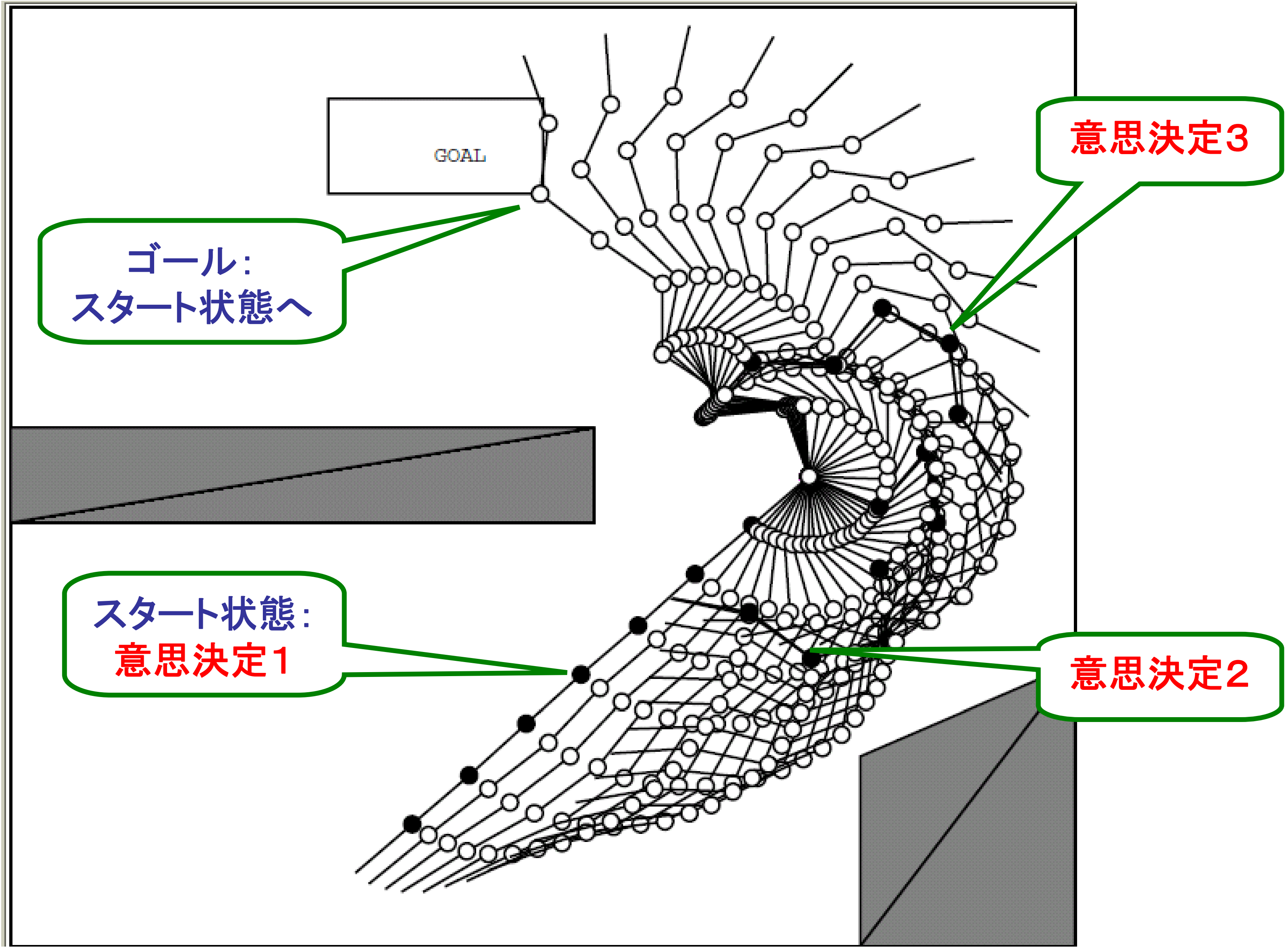


(2)



(3)

動作例



8次元状態×8次元行動の強化学習問題

Q-learning

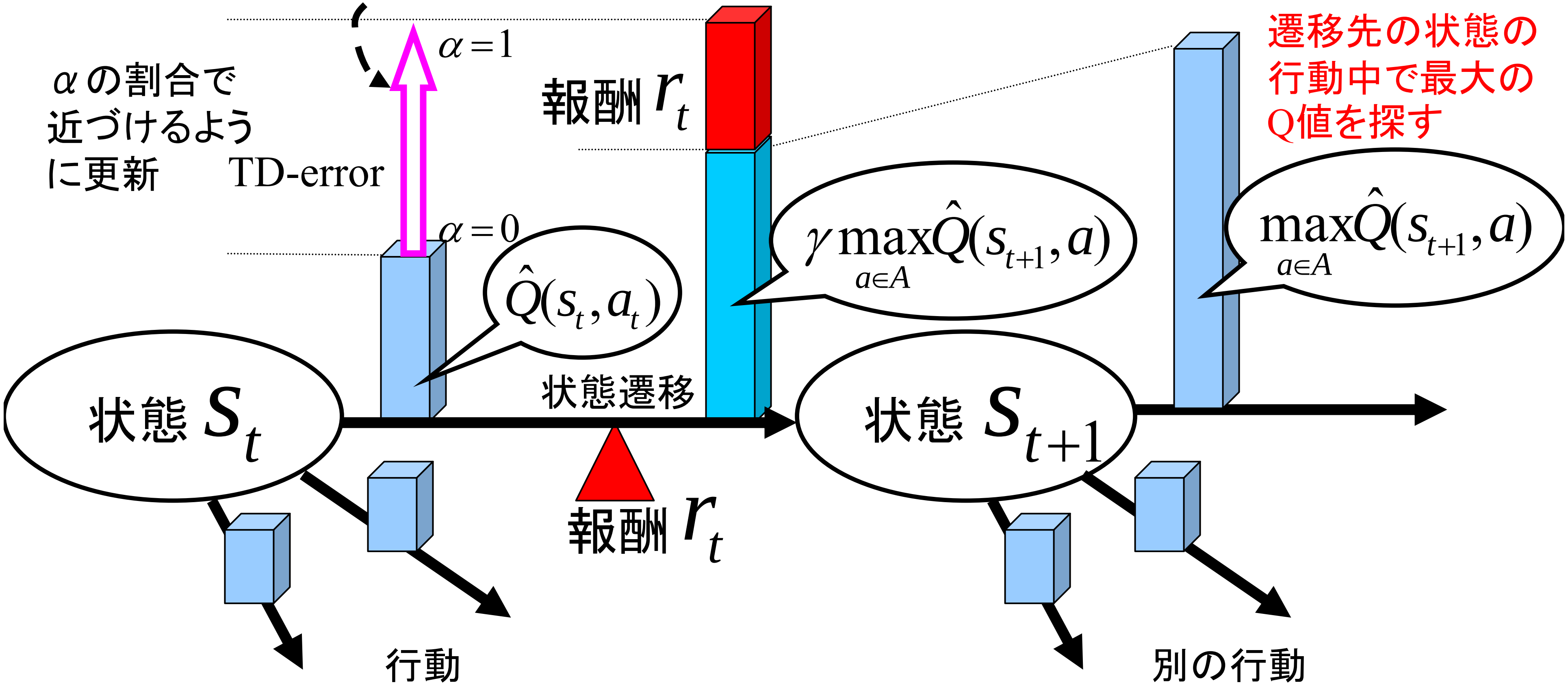
連続空間の関数近似

ボルツマン行動選択

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a \in A} \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t) \right]$$

状態Stで実行した行動のQ値を更新

γ は割引率（$0 \leqq \gamma \leqq 1$）
α は学習率（$0 < \alpha \leqq 1$）



Q-learning | 連続空間の関数近似 | ボルツマン行動選択

Q-learningの収束定理（Watkins92）

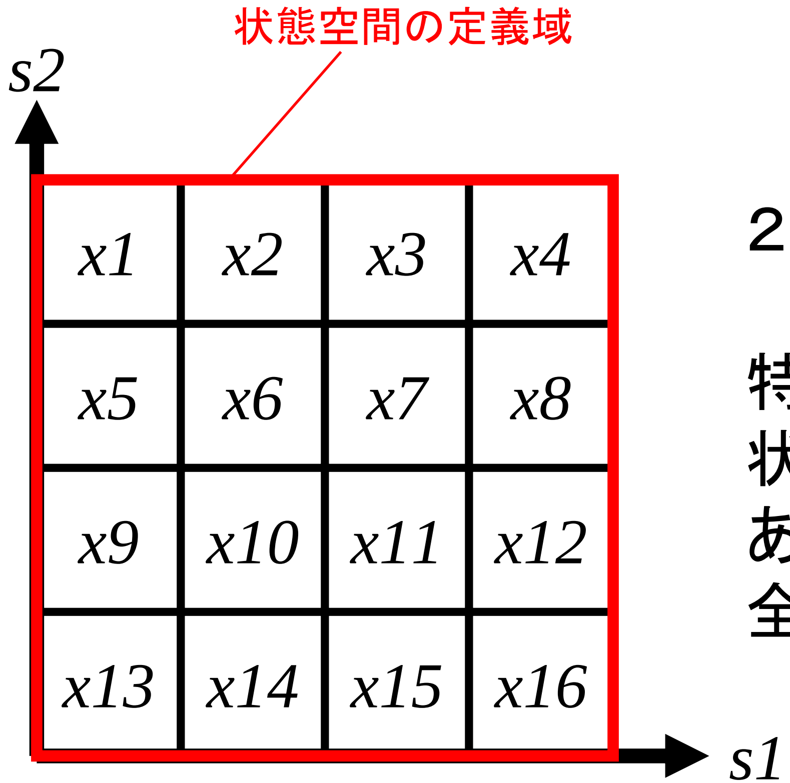
行動選択において全行動を十分な回数選択し，かつ学習率 α が

$$\sum_{t=0}^{\infty} \alpha'(t) \to \infty \qquad \text{and} \qquad \sum_{t=0}^{\infty} \alpha(t)^2 < \infty$$

を満たす時間 t の関数になっているとき，アルゴリズムで得るQ値は確率１で最適政策の評価値に概収束する。

ただし環境はエルゴート性を有するMDP

タイルコーディング（グリッド分割）による特徴ベクトル生成



２次元状態空間 $(s1,s2)$の例

特徴ベクトル $X=(x1,x2, \ldots x16)$
状態が対応する要素xiのタイル内にあるとき $xi = 1$，それ以外の要素は全て 0

この特徴ベクトルを用いたvalue表現と学習における更新規則は離散テーブルを用いた場合と同一になる

線形アーキテクチャにおける更新処理（Q-learning）

強化学習アルゴリズムQ-learning:　以下の式で逐次更新

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha \underbrace{\left[r_t + \gamma \max_{a \in A} \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t) \right]}_{\text{TD-error}}$$

状態Stで実行した
行動のQ値を更新

γは割引率（$0 \leqq \gamma \leqq 1$）
αは学習率（$0 < \alpha \leqq 1$）

線形アーキテクチャでは特徴ベクトルxと重みwで表される

$$\hat{Q}(S, a) = x_1 w_1 + x_2 w_2 + x_3 w_3$$

以下の式で重みwを更新する:

$$w_i \leftarrow w_i + \alpha \ \mathrm{TDerror} \boxed{\frac{\partial \hat{Q}(S, a)}{\partial w_i}}$$

x_i

線形アーキテクチャの更新法では真のvalueに対して二乗誤差最小の近似へ収束することが証明されている(Bertsekas & Tsitsiklis 96)　ただし$\alpha \rightarrow 0$

線形アーキテクチャによる汎化と関数近似

望ましい特徴ベクトル $x_1 \ x_2 \ x_3$ の性質:
絶対和ノルム＝1 （ただし収束の必要条件ではない）

異なる状態間では互いに線形独立

特徴ベクトルの個数を増やす → 近似性能（精度）向上
特徴量のカバーする領域を広げる→ 汎化（補間）性能向上

特徴ベクトルの生成方法には数多くのバリエーションがある

- ●離散状態表現（タイルコーディング）＝線形アーキテクチャの一種
- ●CMAC（Sutton98）:タイルコーディングを複数組み合わせ
- ●高次元の空間:CMAC＋ハッシュ

なぜ強化学習では線形アーキテクチャなのか？

非線形の関数近似（ニューラルネット等）を用いると，
収束は保証されず，学習が発散する例も示されている（Baird95）

Q-learning | **連続空間の関数近似** | ボルツマン行動選択

状態入力（センサ入力） → 固定変換 → **高次元の特徴量ベクトル** $(f_1, f_2, \cdots f_n)$

e.g., grid coding,
tile coding CMAC, etc.

それぞれが基底関数の値に対応

線形アーキテクチャ $Q(s,a) = \sum_{i=1}^{n} f_i w_i$

各特徴量に対応する重み変数

等しい

Q-learning 更新規則

$$w_i \leftarrow w_i + \alpha \,\mathrm{TDerror} \boxed{\frac{\partial \hat{Q}(S,a)}{\partial w_i}}$$

収束定理が存在 (Tsitsiklis and Van Roy, 1997)

Q-learning

連続空間の関数近似

ボルツマン行動選択

状態入力（センサ入力）

固定変換

e.g., grid coding, tile coding CMAC, etc.

高次元の特徴量ベクトル

$$\left(f_1, f_2, \cdots f_n\right)$$

それぞれが基底関数の値に対応

状態間で線形独立な特徴ベクトルをどのように生成するか？

線形アーキテクチャ

$$Q(s,a) = \sum_{i=1}^{n} f_i \, w_i$$

各特徴量に対応する重み変数

Q-learning

連続空間の関数近似

ボルツマン行動選択

状態入力（センサ入力）

固定変換

e.g., grid coding,
tile coding CMAC, etc.

高次元の特徴量ベクトル
$$\left(f_1, f_2, \cdots f_n\right)$$

それぞれが基底関数の値に対応

線形アーキテクチャ

$$Q(s,a) = \sum_{i=1}^{n} f_i \, w_i$$

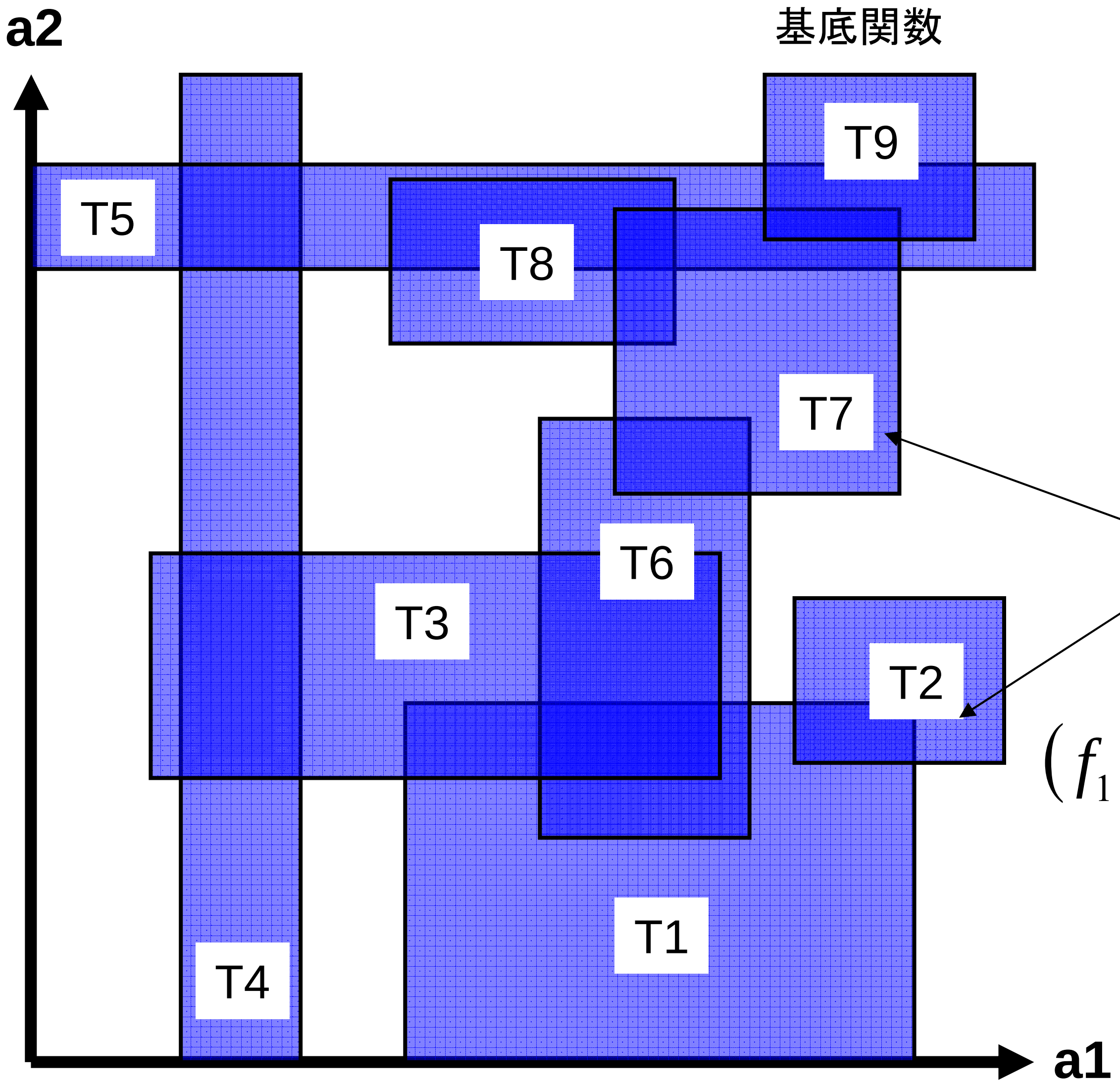
各特徴量に対応する重み変数

多様な非線形変換を用いて膨大な特徴量ベクトルを生成すると良い

ランダムな特徴は効果的!

例）　サポートベクターマシン

高次元状態・行動空間の汎化：ランダムタイリングによるQ関数表現



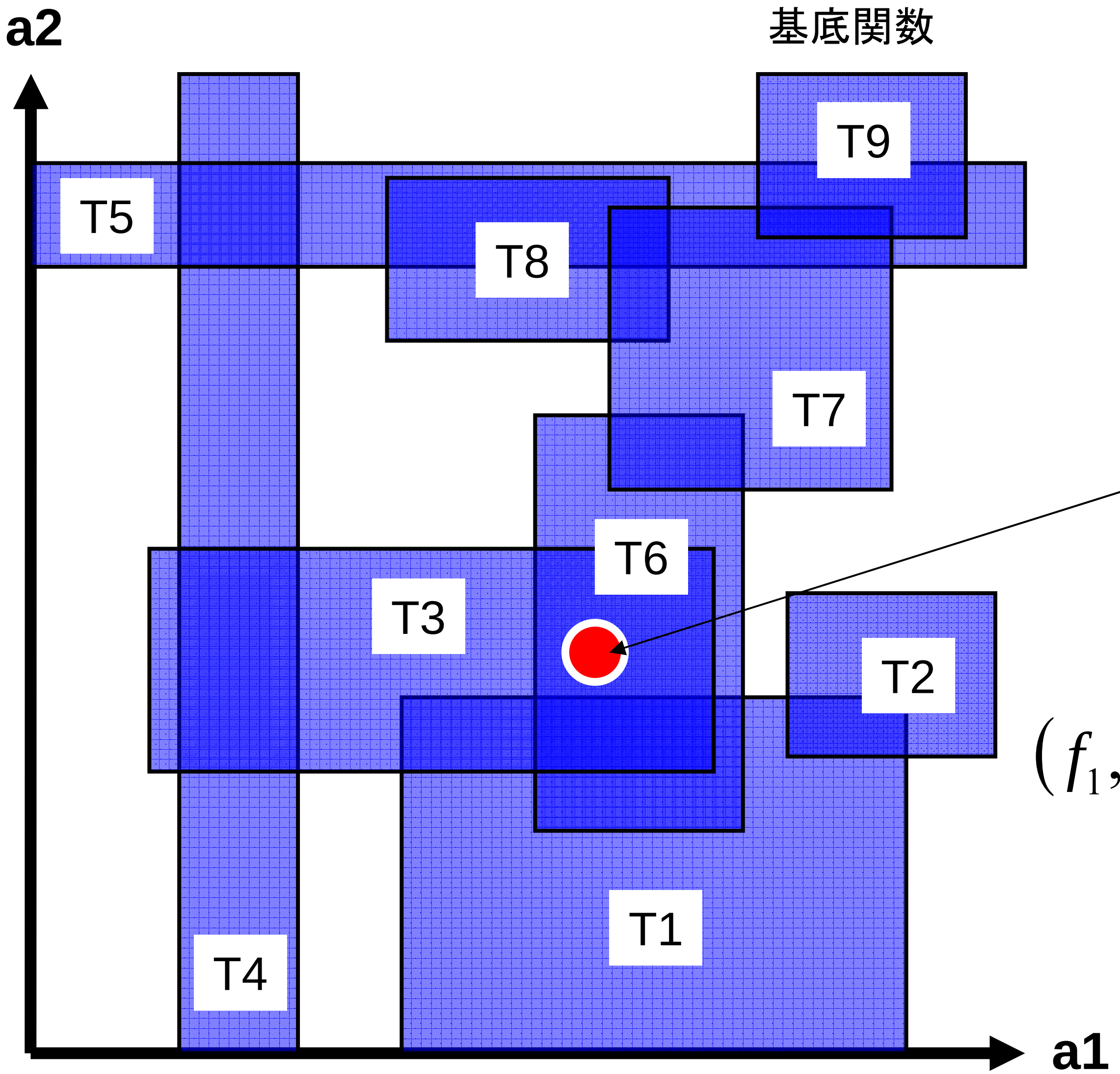
様々な大きさ・次元の超矩形タイルをランダムに配置して特徴量ベクトルを生成

各タイルが特徴量ベクトルの各要素に対応

$$\left(f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9\right)$$

実験では、一様分布入力に対してヒットする確率が0.16になるよう領域を設定

高次元状態・行動空間の汎化：
ランダムタイリングによるQ関数表現

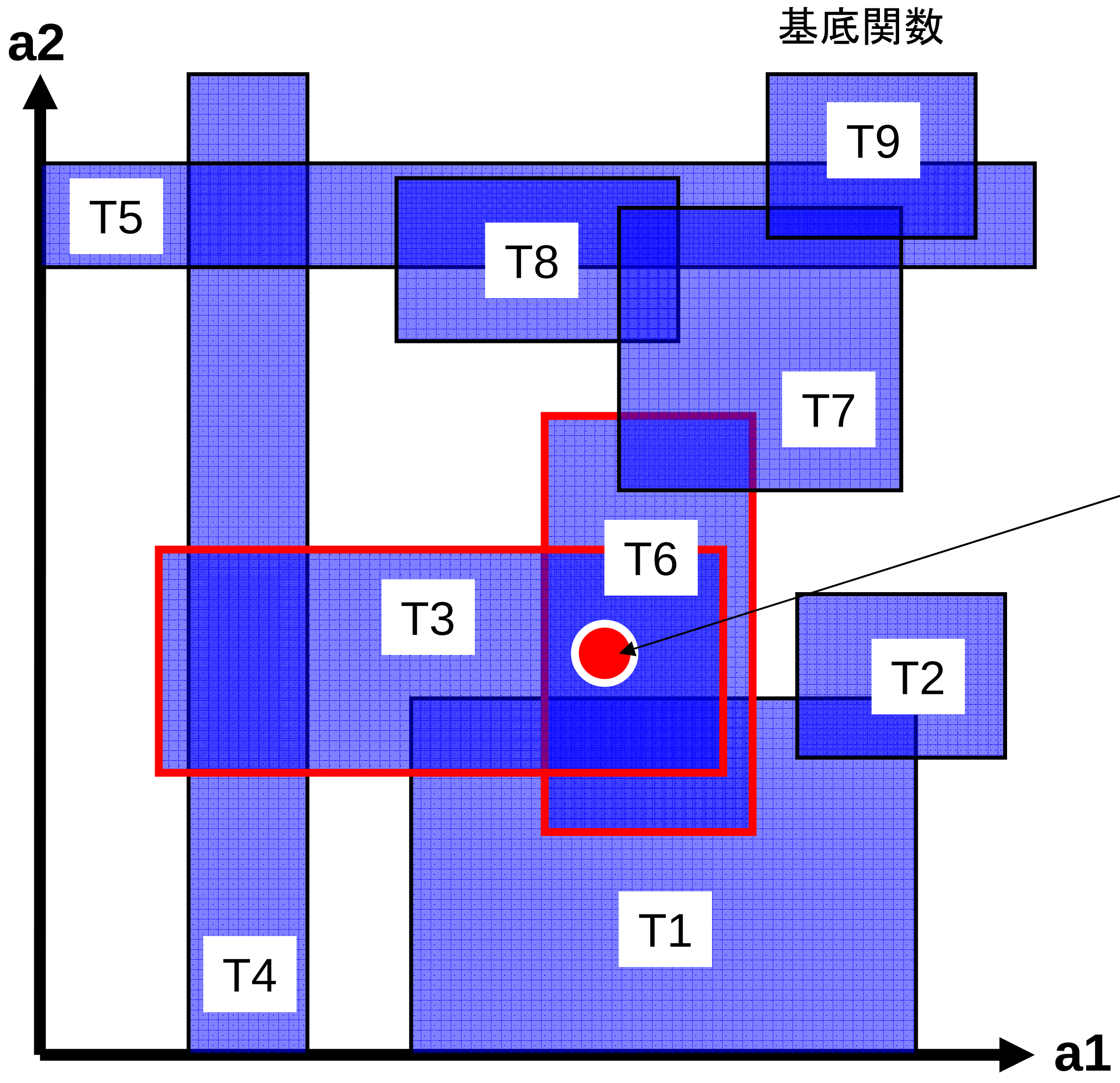


様々な大きさ・次元の超矩形タイルをランダムに配置して特徴量ベクトルを生成

ある値が入力されたとき

$$\left(f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9\right)$$

高次元状態・行動空間の汎化：
ランダムタイリングによるQ関数表現



様々な大きさ・次元の超矩形タイルをランダムに配置して特徴量ベクトルを生成

ある値が入力されたとき

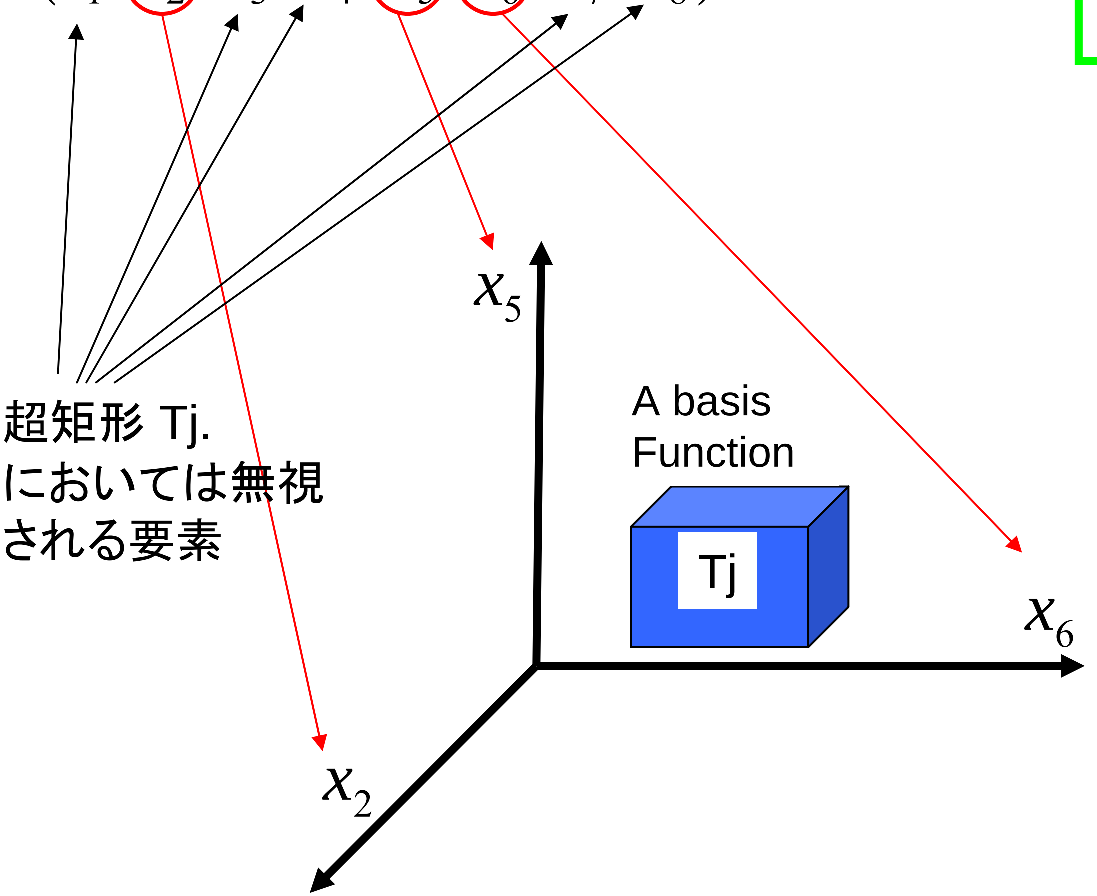
$$(0, 0, 1, 0, 0, 1, 0, 0, 0)$$

該当するタイルの特徴量のみ1

高次元空間における汎化:
Random-rectangular Coarse Coding

入力ベクトル

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$$



ランダムに生成された超矩形

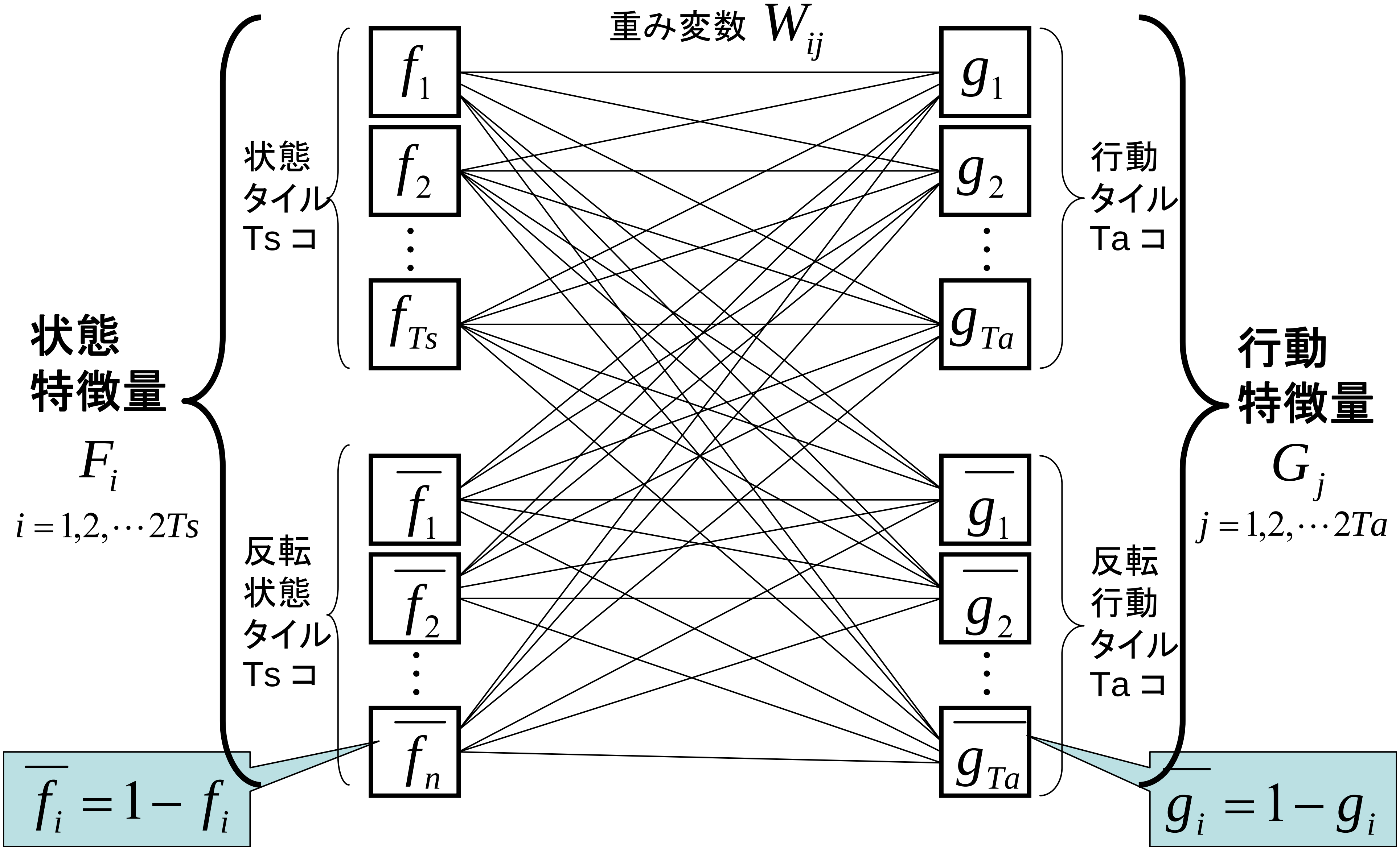
様々なサイズと次元で定義

各超矩形においては、
矩形を定義するベクトル要素の
位置と数はランダムに決められる
.

この例では入力ベクトルは8次元、
しかし特徴量を表す基底となる
超矩形は x_2, x_5 ,x_6 のみで定義
され、他の要素は無視される

ある状態 s, 行動 a
におけるQ値の求め方

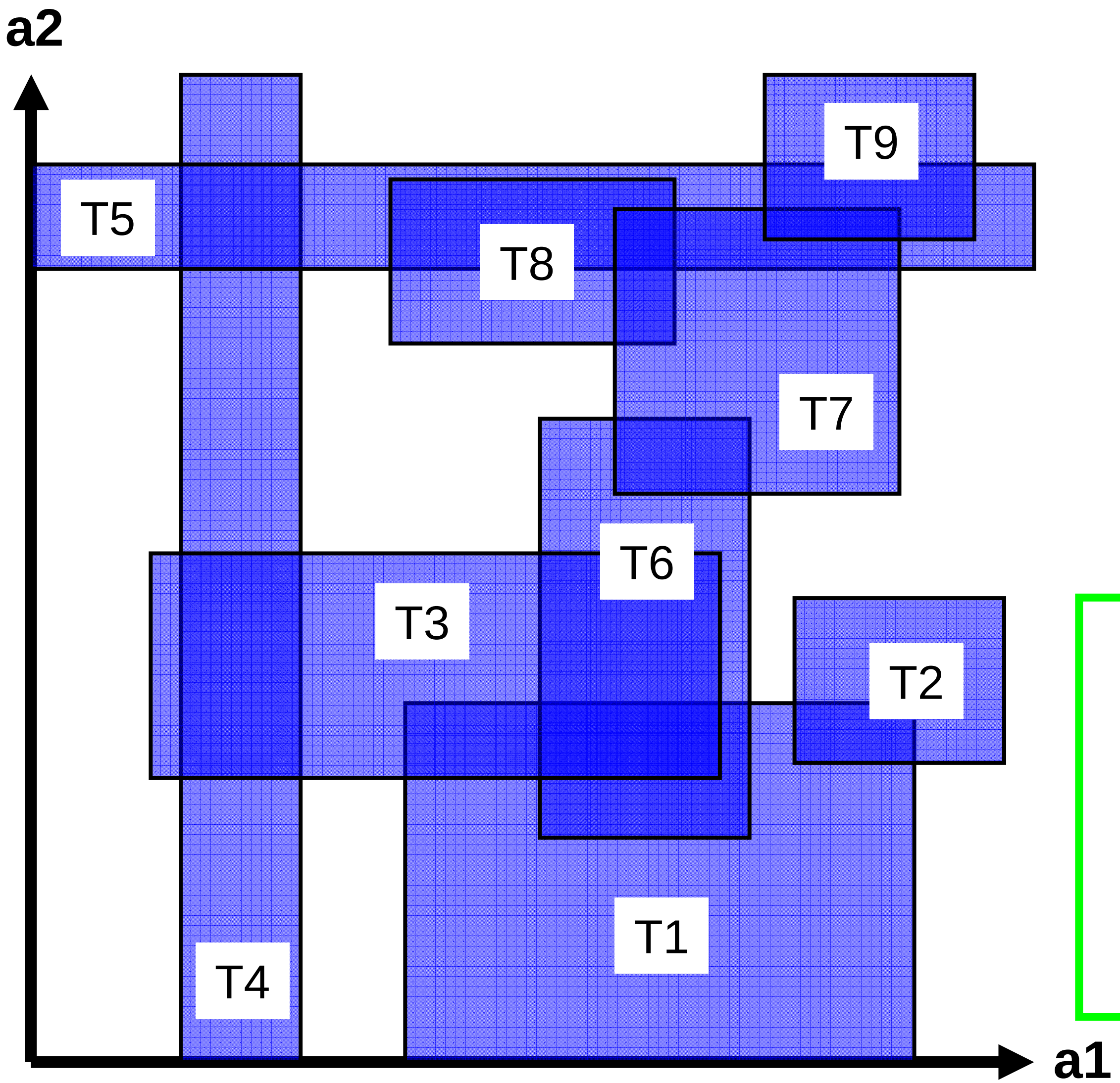
$$Q(s,a) = \frac{1}{Ts\,Ta} \sum_{j=1}^{2Ta} \sum_{i=1}^{2Ts} F_i \; G_j \; W_{ij}$$



Q-learning

連続空間の関数近似

ボルツマン行動選択



ランダムタイリングによる高次元連続空間の汎化

特長

- 実装が簡単
- 空間爆発を回避
- 足りなければタイルを増やすだけ
- 実装が問題に依存しない

シミュレーションによる関数近似性能の定量的評価

Q-関数で**2つの入力を区別可能** ＝ 2つの入力に対応する特徴量ベクトルが互いに**線形独立**

2～8次元
入力ベクトル

任意に2つ
ランダムに発生

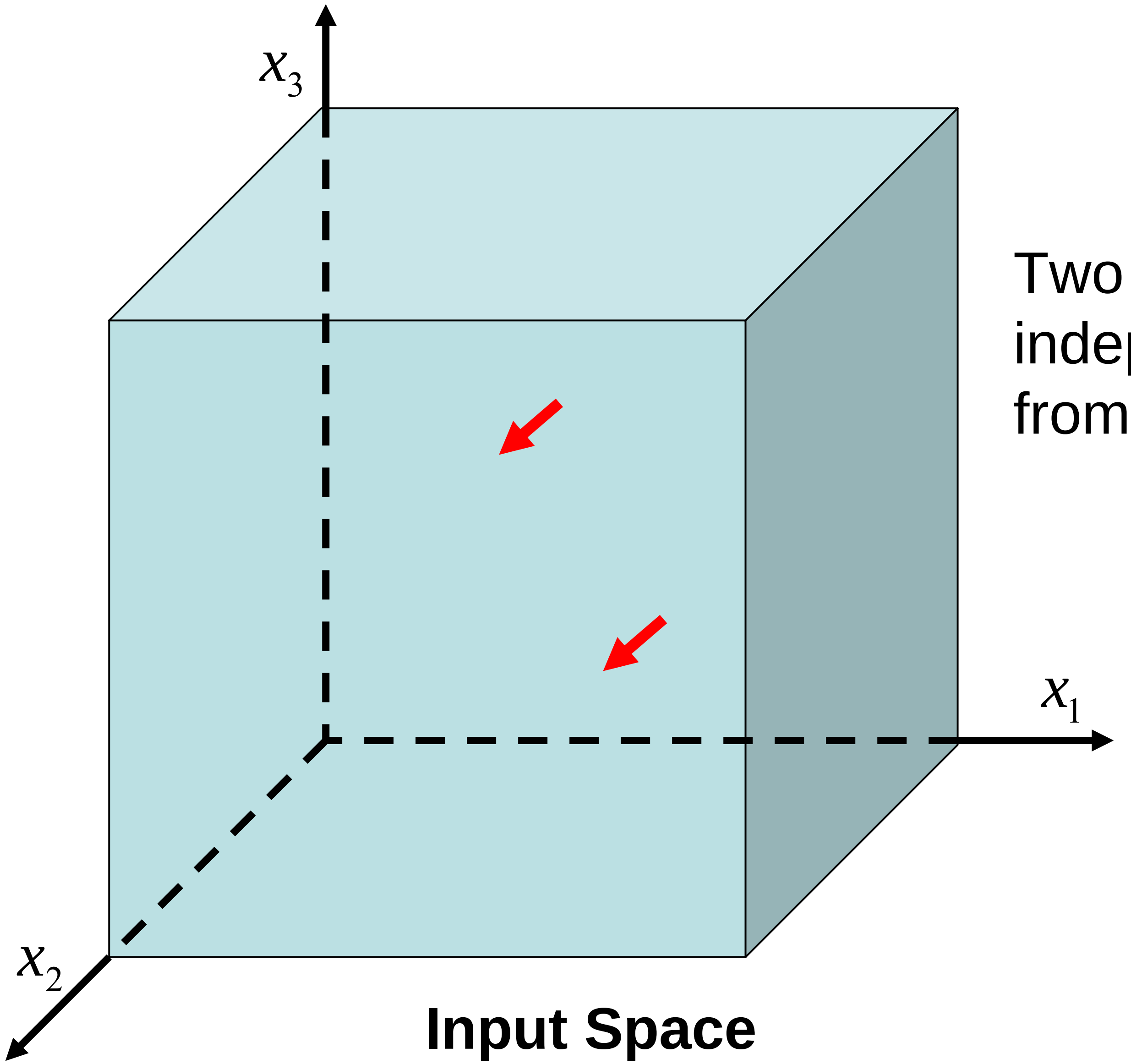
Random Rectangular
Coarse Coding

高次元
特徴量ベクトル
$$\left(f_1, f_2, \cdots f_n\right)$$

入力に対応する
2つの特徴量ベ
クトルが互いに
独立か？

1000種類のランダムな入力ベクトルペアを生成し、
提案手法にて特徴量ベクトルを生成させ、
それらが線形独立になっているかどうかの割合を調べる

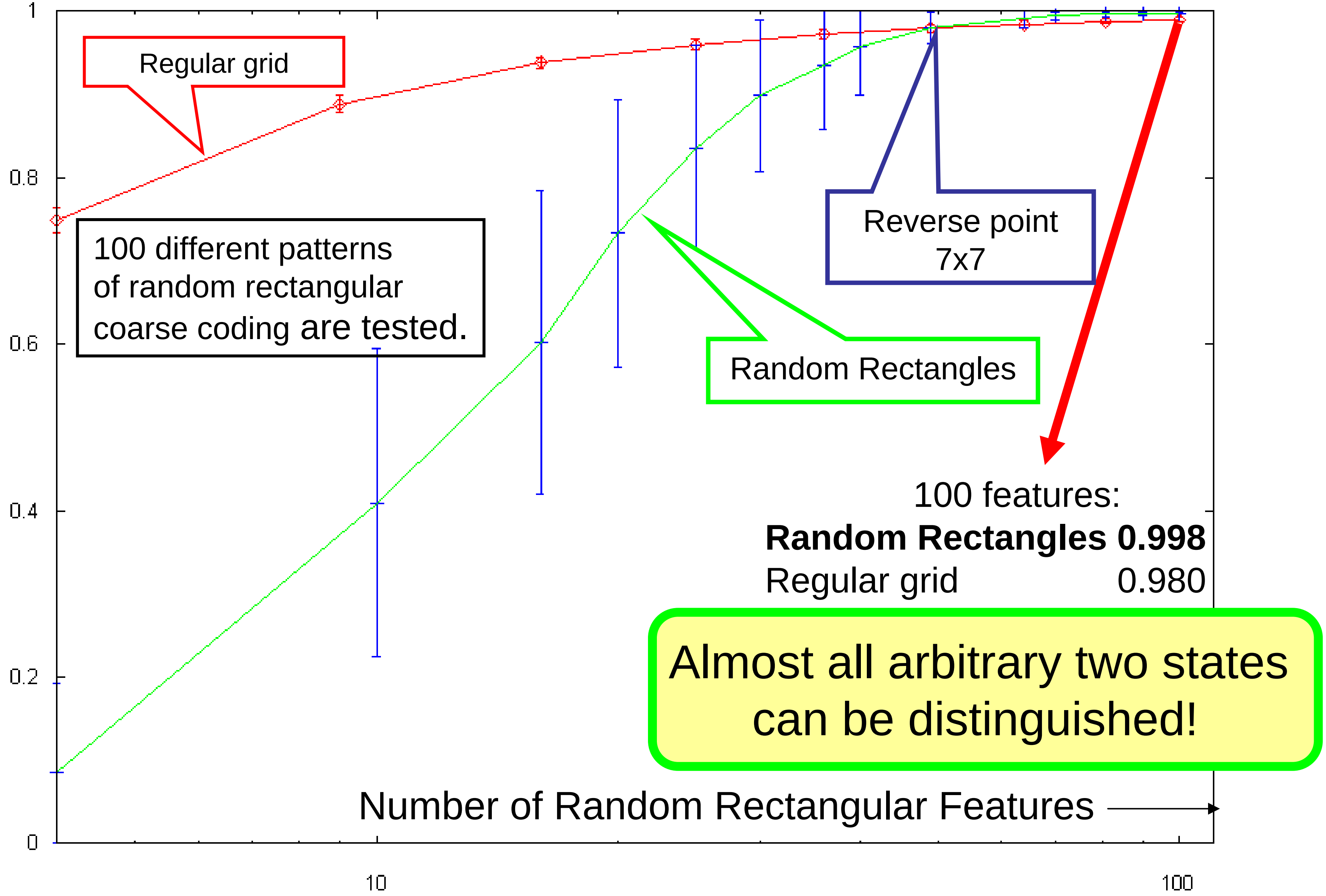
Condition of input vector:



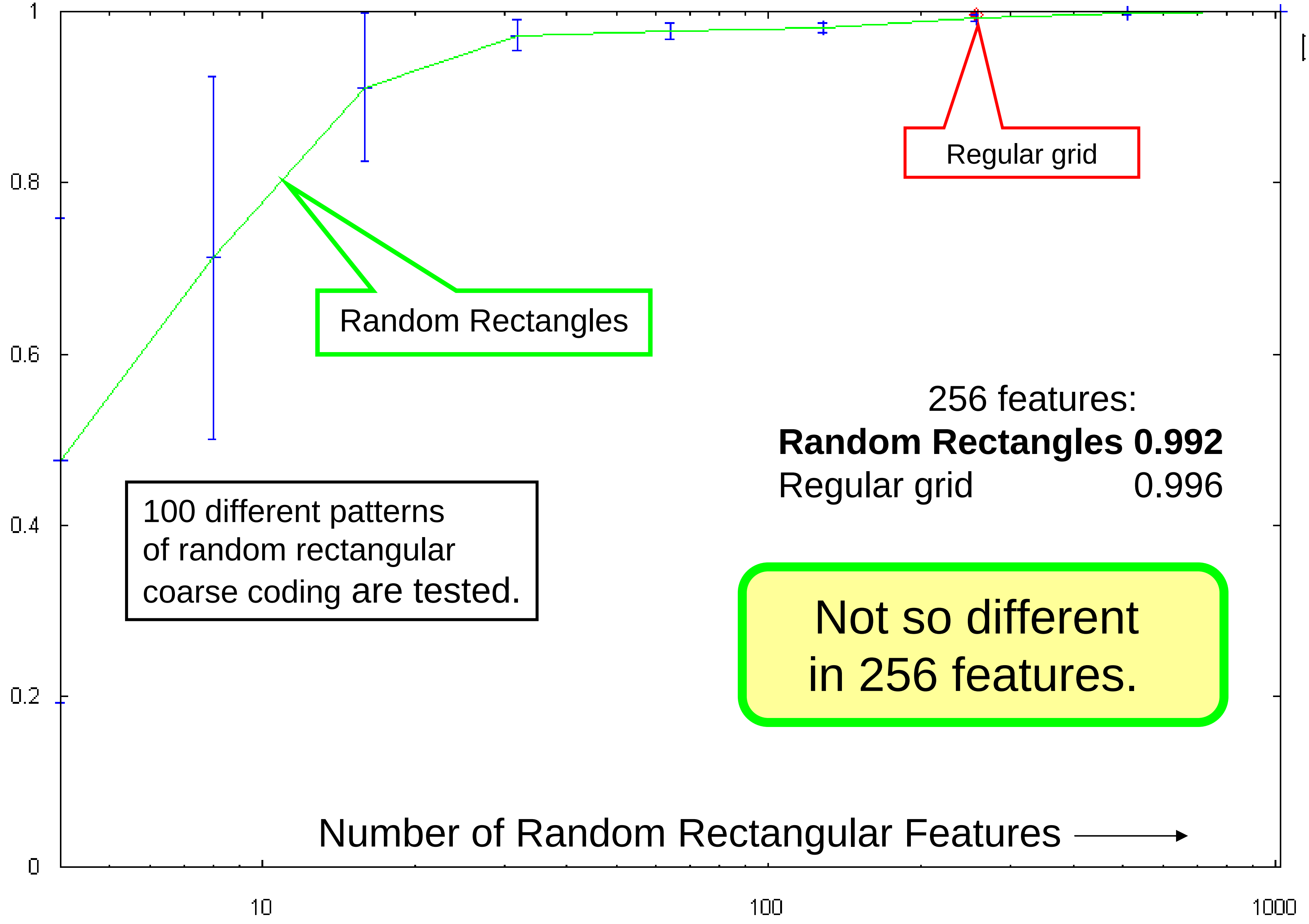
Two input vectors are independently sampled from Uniform distribution.

Input Space

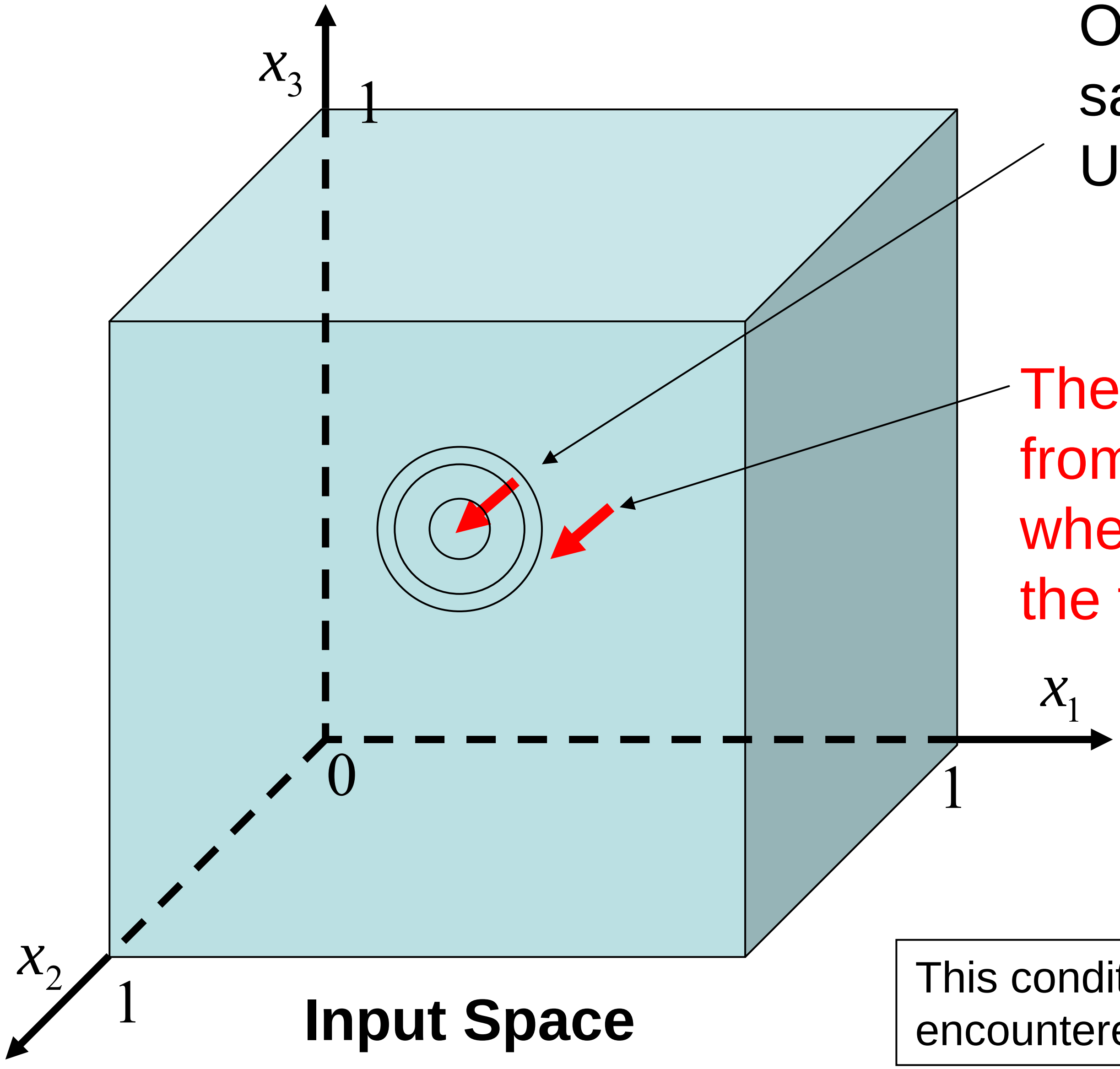
Linearly Independent Rate of Feature Vectors between two Random Vectors in **2-dimensional** Input Space



Linearly Independent Rate of Feature Vectors
between two Random Vectors in **8-dimensional** Input Space



Another condition of input vector:



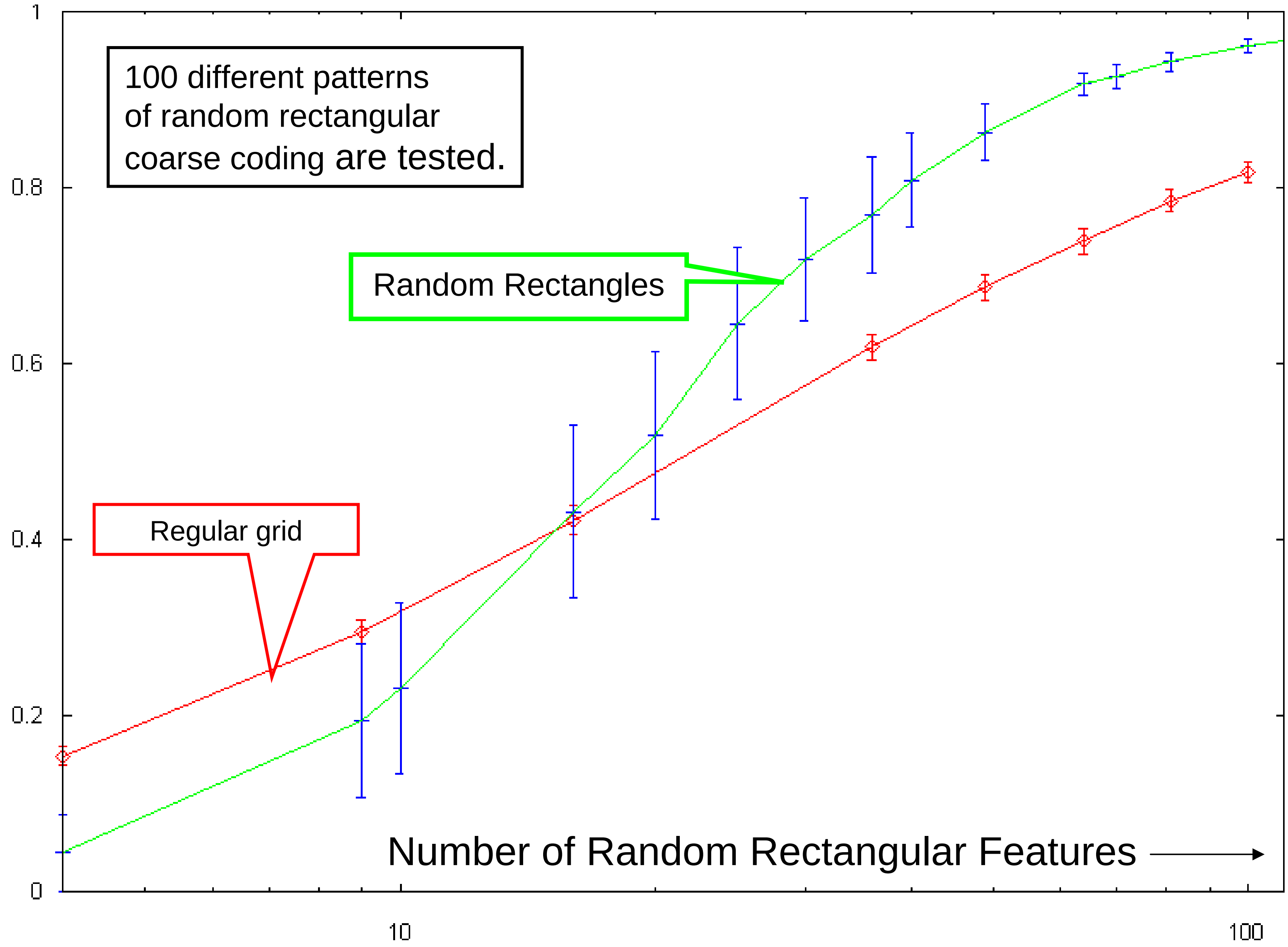
One input vector is sampled from Uniform distribution.

The other is sampled from normal distribution where the center is to the former selected point.

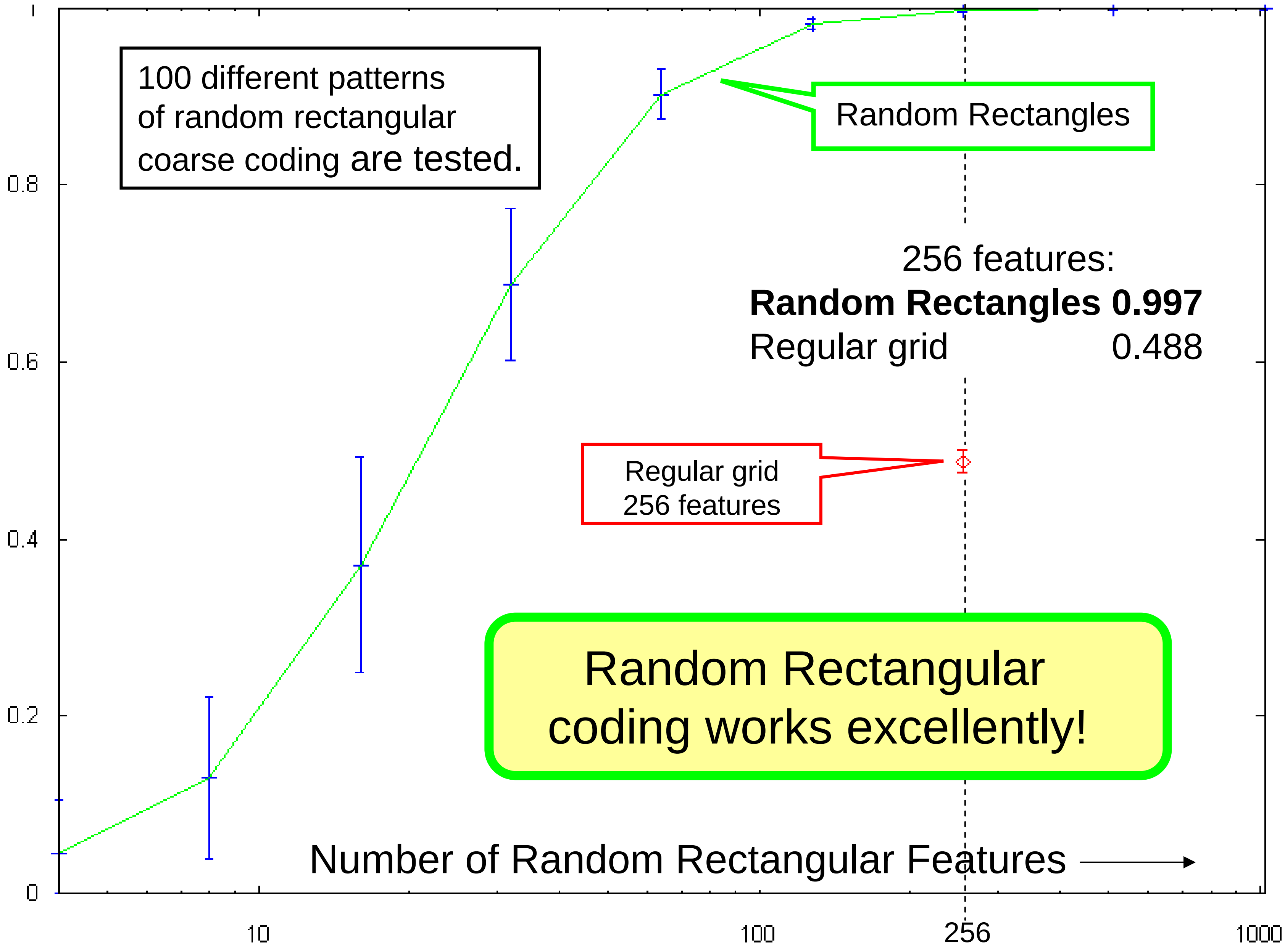
Deviation of the normal distribution is 0.1,
And covariance is 0.

This condition is quite often encountered in RL in Robotics.

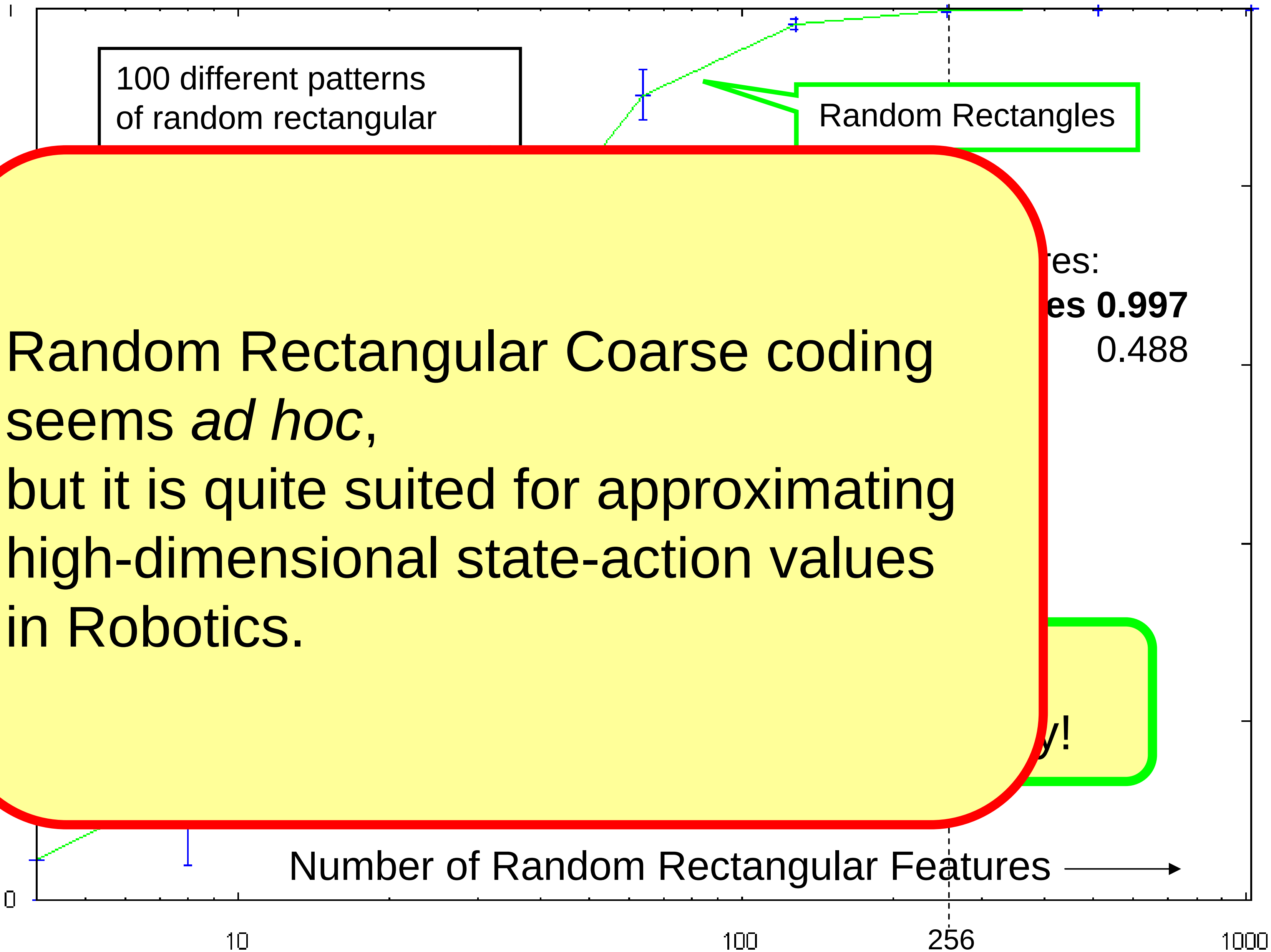
Linearly Independent Rate of Feature Vectors
between two Random Vectors in **2-dimensional** Input Space



Linearly Independent Rate of Feature Vectors
between two Random Vectors in **8-dimensional** Input Space



Linearly Independent Rate of Feature Vectors
between two Random Vectors in **8-dimensional** Input Space

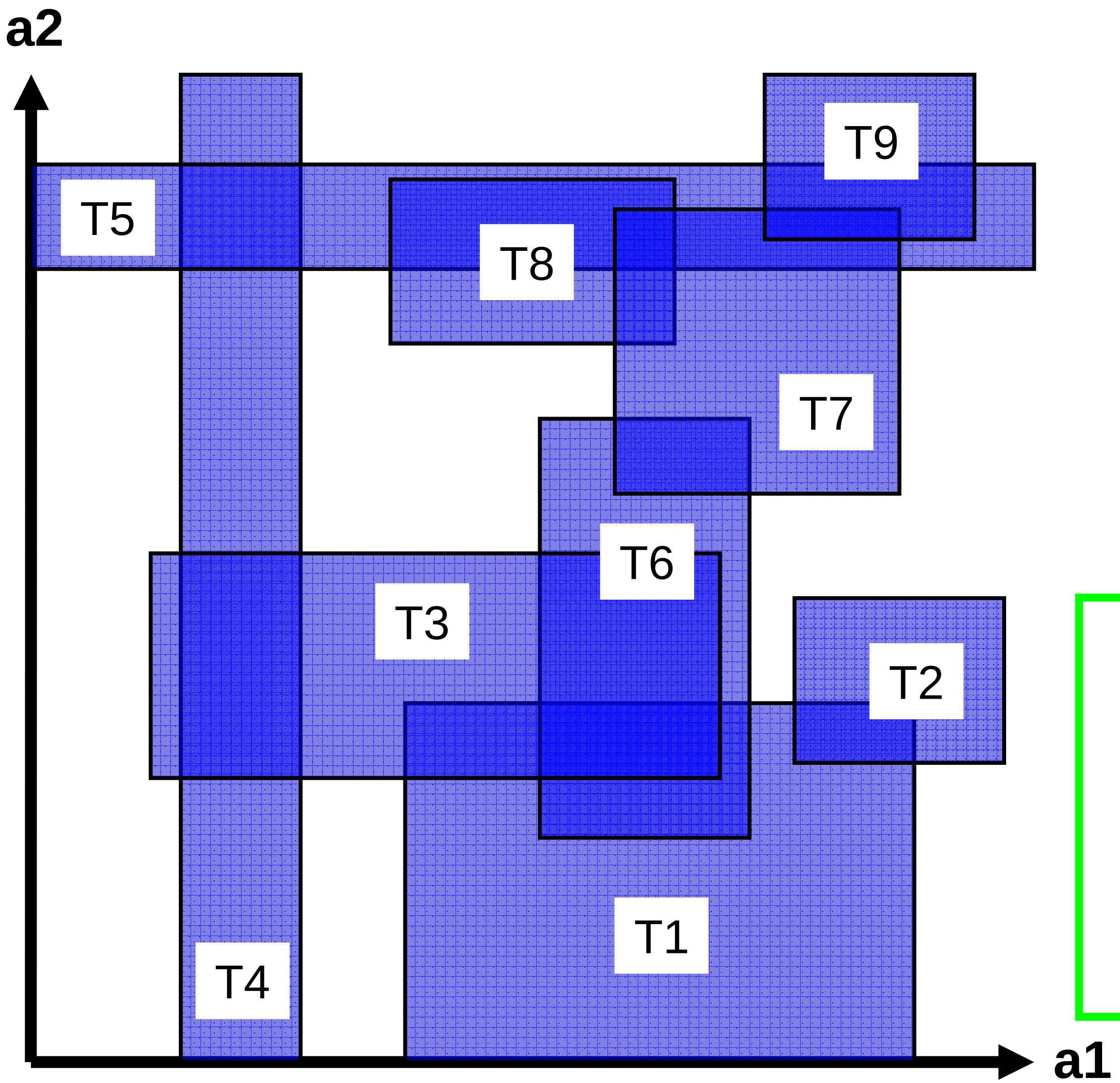


Random Rectangular Coarse coding seems *ad hoc*,
but it is quite suited for approximating high-dimensional state-action values in Robotics.

Q-learning

連続空間の関数近似

ボルツマン行動選択



ランダムタイリングによる高次元連続空間の汎化

特長

- 実装が簡単
- 空間爆発を回避
- 足りなければタイルを増やすだけ
- 実装が問題に依存しない

Q-learning

連続空間の関数近似

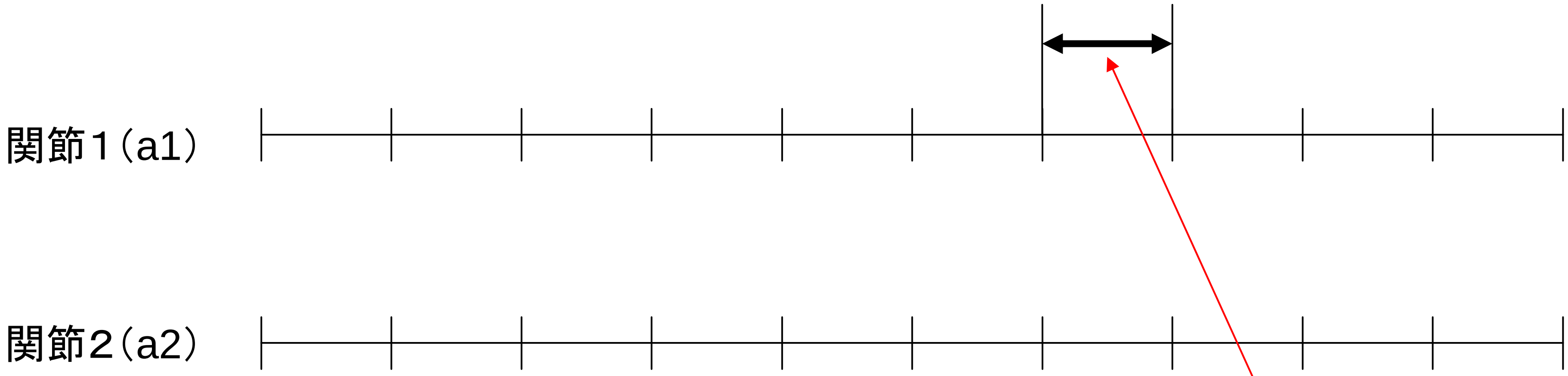
ボルツマン行動選択

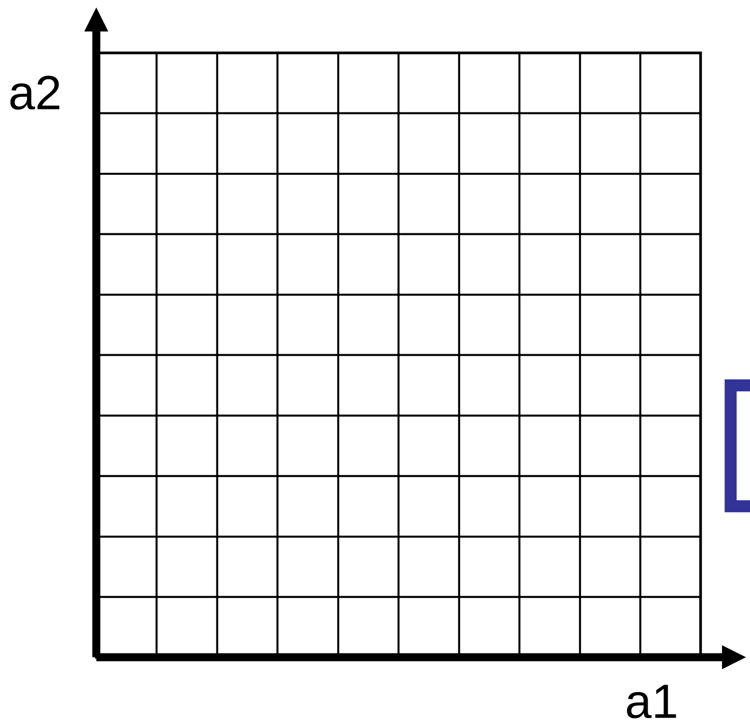
ある状態SにおいてQ値のボルツマン分布に従って
確率的に行動aを選ぶ

行動選択確率：
$$P(a_i) = \frac{\exp\left(\frac{Q(s,a_i)}{T}\right)}{\sum_{j=1}^{A} \exp\left(\frac{Q(s,a_j)}{T}\right)}$$

しかし！
全行動空間の exp(Q) の合計が必要！

高次元行動空間の扱いの難しさ：行動選択





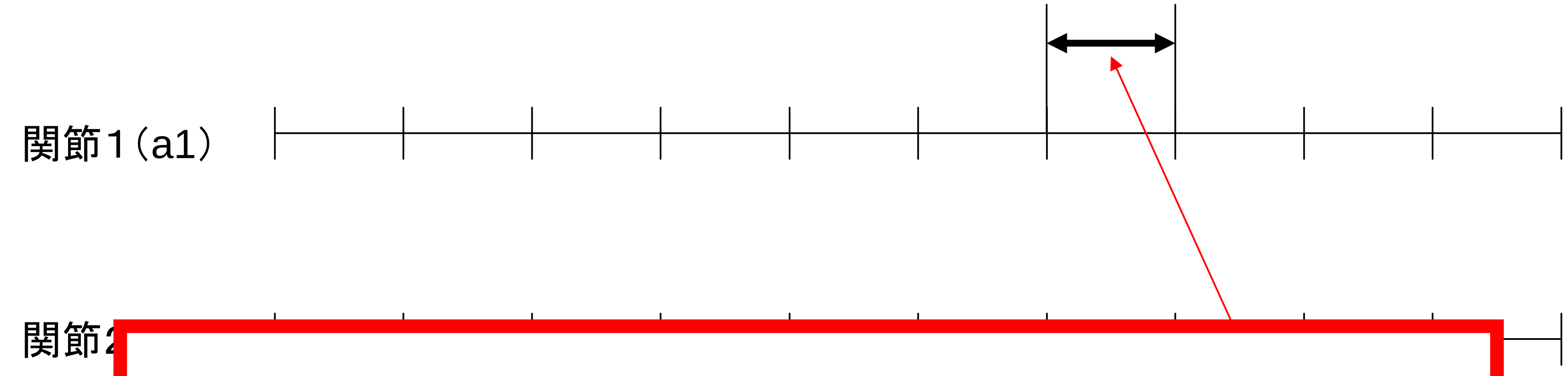
行動空間の各次元を<u>10分割で離散化</u>

8関節　→　離散行動　1000万通り

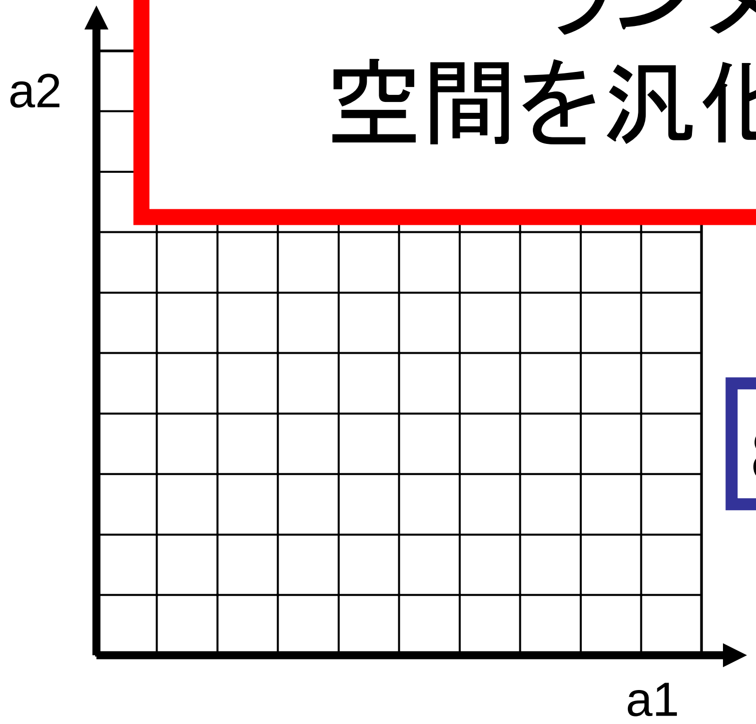
Q-learningにおける行動選択やQ値の更新の際に最大のQ値を探す計算に多大な計算コストを要する

従来の計算コストは「次元の呪い」に直面

高次元行動空間の扱いの難しさ：行動選択



ランダムタイリングで
空間を汎化しても意味がない!?



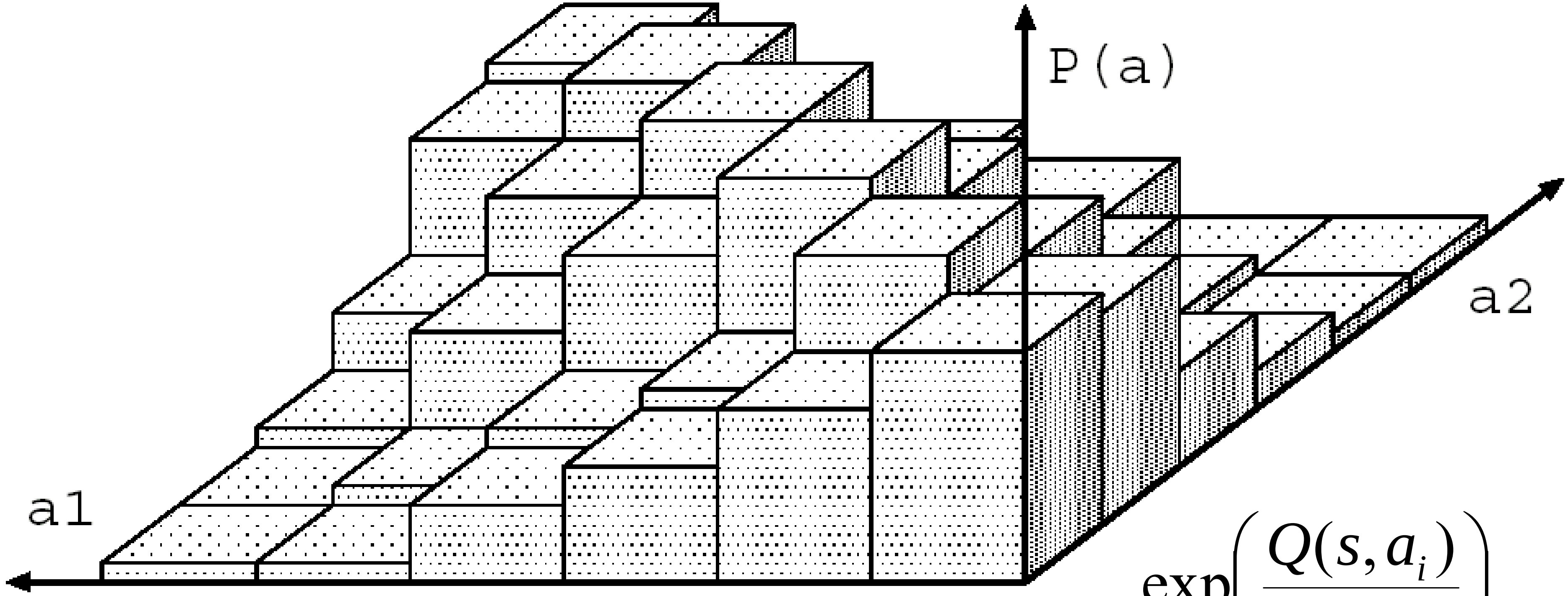
8関節 → 離散行動 1000万通り

Q-learningにおける行動選択やQ値の更新の際に
最大のQ値を探す計算に多大な計算コストを要する
従来の計算コストは「次元の呪い」に直面

Q-learning

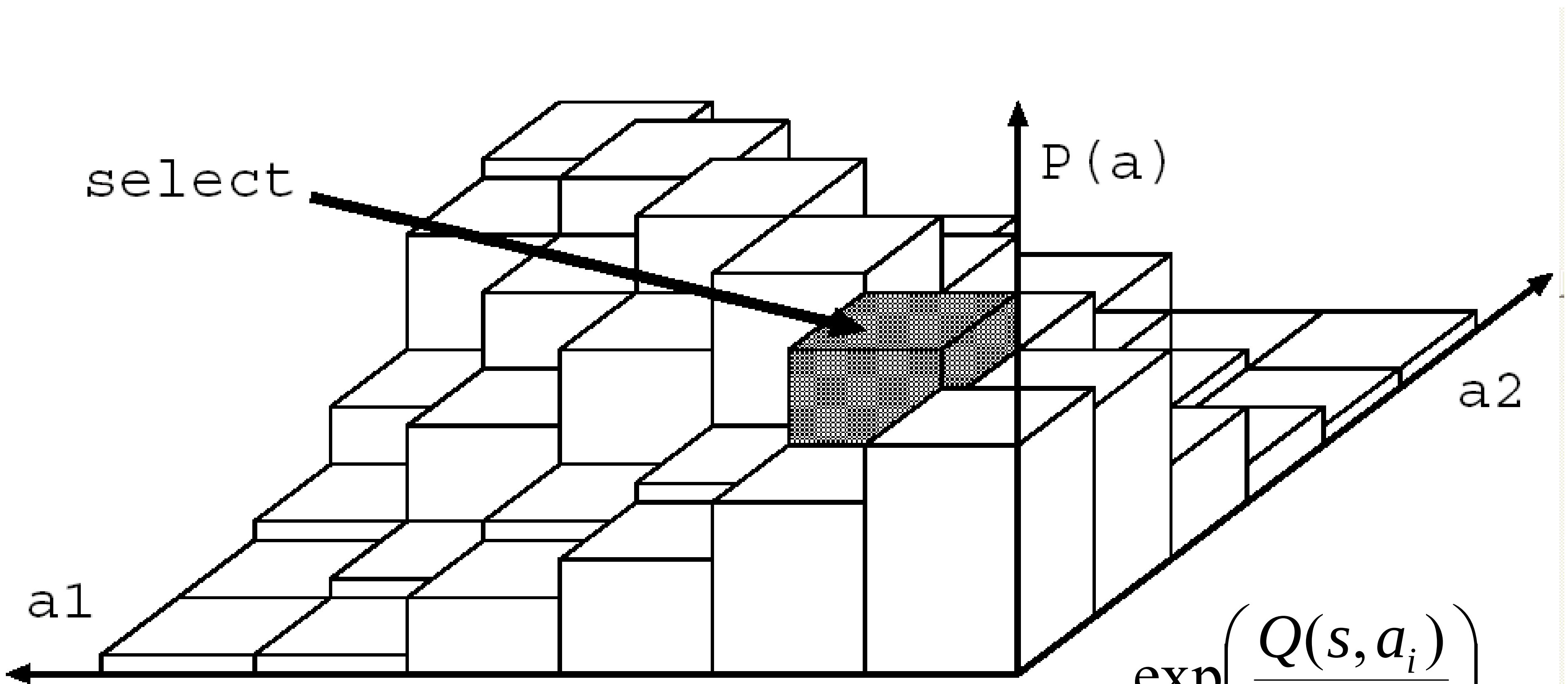
連続空間の関数近似

ボルツマン行動選択



$$P(a_i) = \frac{\exp\left(\frac{Q(s,a_i)}{T}\right)}{\sum_{j=1}^{A} \exp\left(\frac{Q(s,a_j)}{T}\right)}$$

Q-learning | 連続空間の関数近似 | **ボルツマン行動選択**



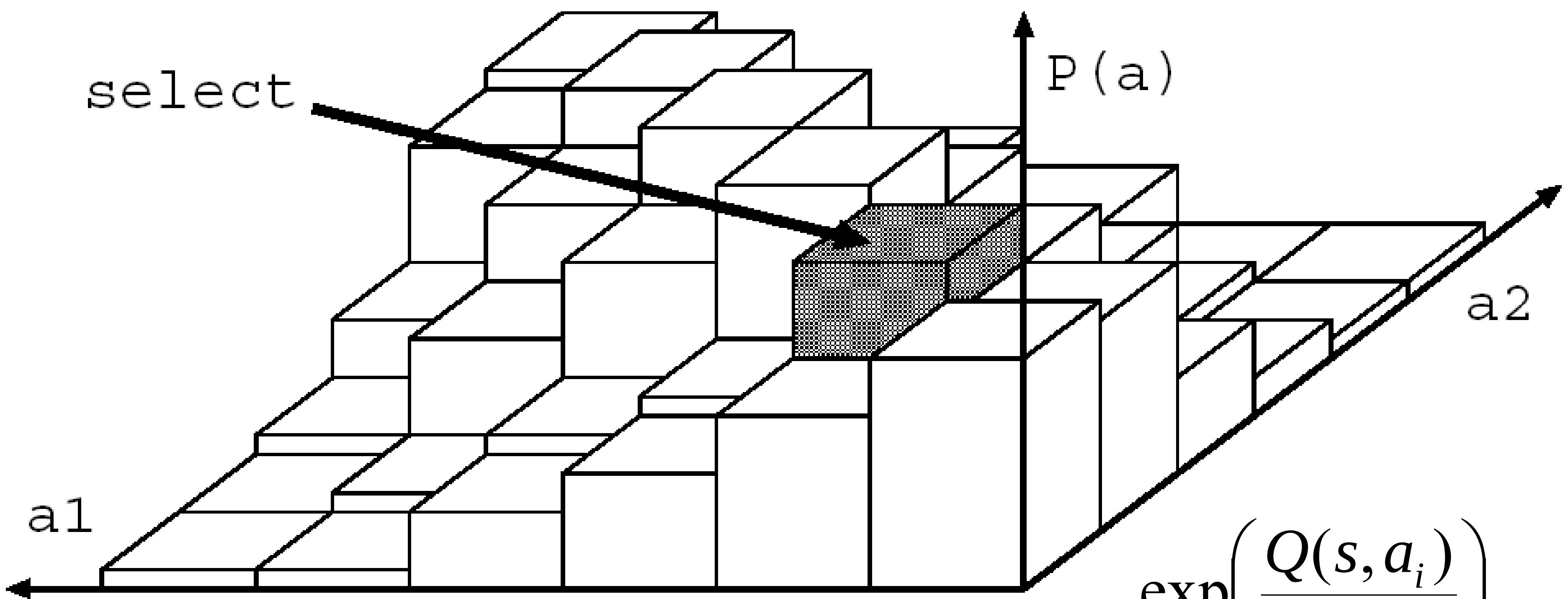
$$P(a_i) = \frac{\exp\left(\dfrac{Q(s,a_i)}{T}\right)}{\sum_{j=1}^{A} \exp\left(\dfrac{Q(s,a_j)}{T}\right)}$$

全行動空間の exp(Q)
の合計が必要！

Q-learning

連続空間の関数近似

ボルツマン行動選択



$$P(a_i) = \frac{\exp\left(\frac{Q(s,a_i)}{T}\right)}{\sum_{j=1}^{A} \exp\left(\frac{Q(s,a_j)}{T}\right)}$$

全行動空間の exp(Q)
の合計が必要！

Q-learning

連続空間の関数近似

ボルツマン行動選択

高次元空間における確率分布に従ったサンプルを効率よく得るには？　→　統計学的な一般問題

Markov chain Monte-Carlo（MCMC）法の一種

Gibbsサンプリング

計算が簡単な1次元の条件付確率分布

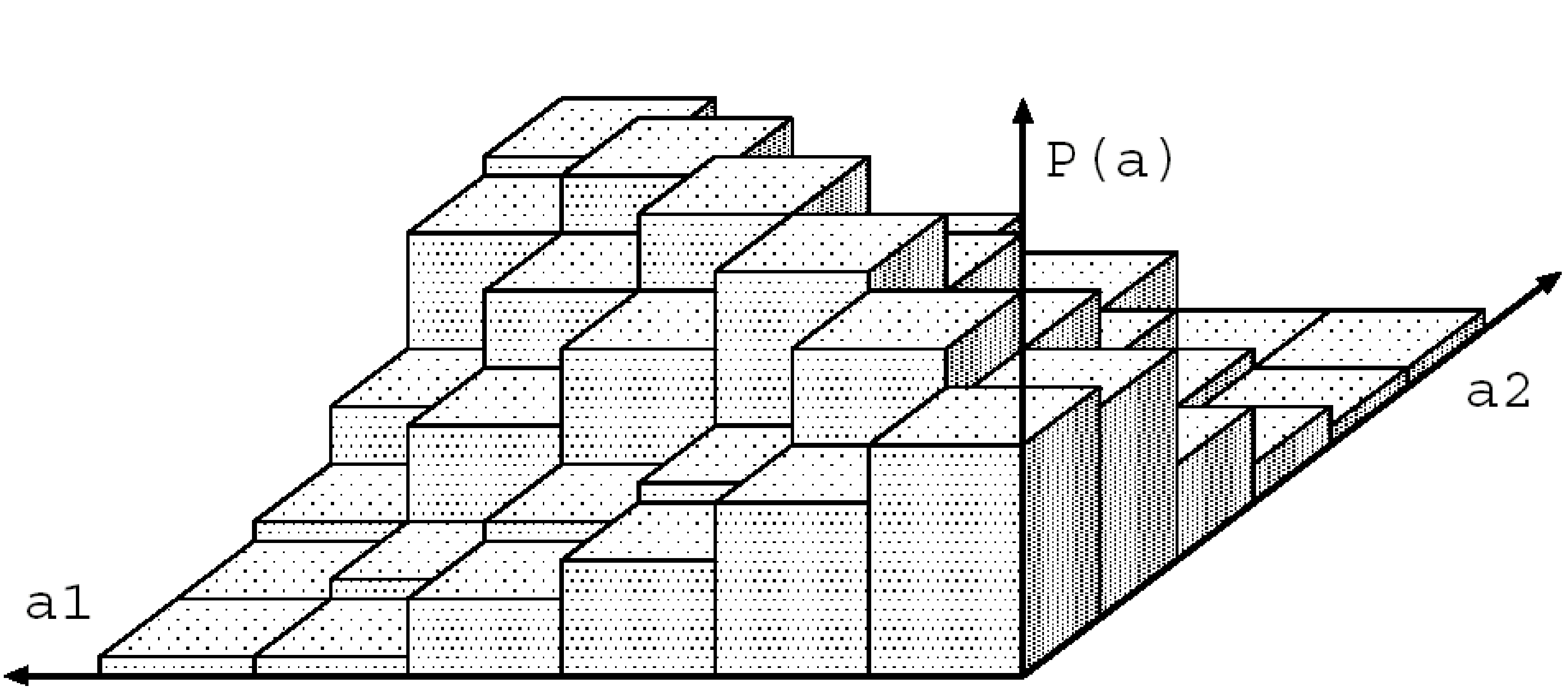
$$
\left.
\begin{aligned}
a^1(t+1) &\approx P(a^1 \mid a^2(t), a^3(t), \cdots a^N(t)) \\
a^2(t+1) &\approx P(a^2 \mid a^1(t), a^3(t), \cdots a^N(t)) \\
&\vdots \\
a^N(t+1) &\approx P(a^N \mid a^1(t), a^2(t), \cdots a^{N-1}(t))
\end{aligned}
\right\}
$$

このような反復 t を十分な回数繰返し、最終的に得たa(t)をサンプルとする

Q-learning

連続空間の関数近似

ボルツマン行動選択

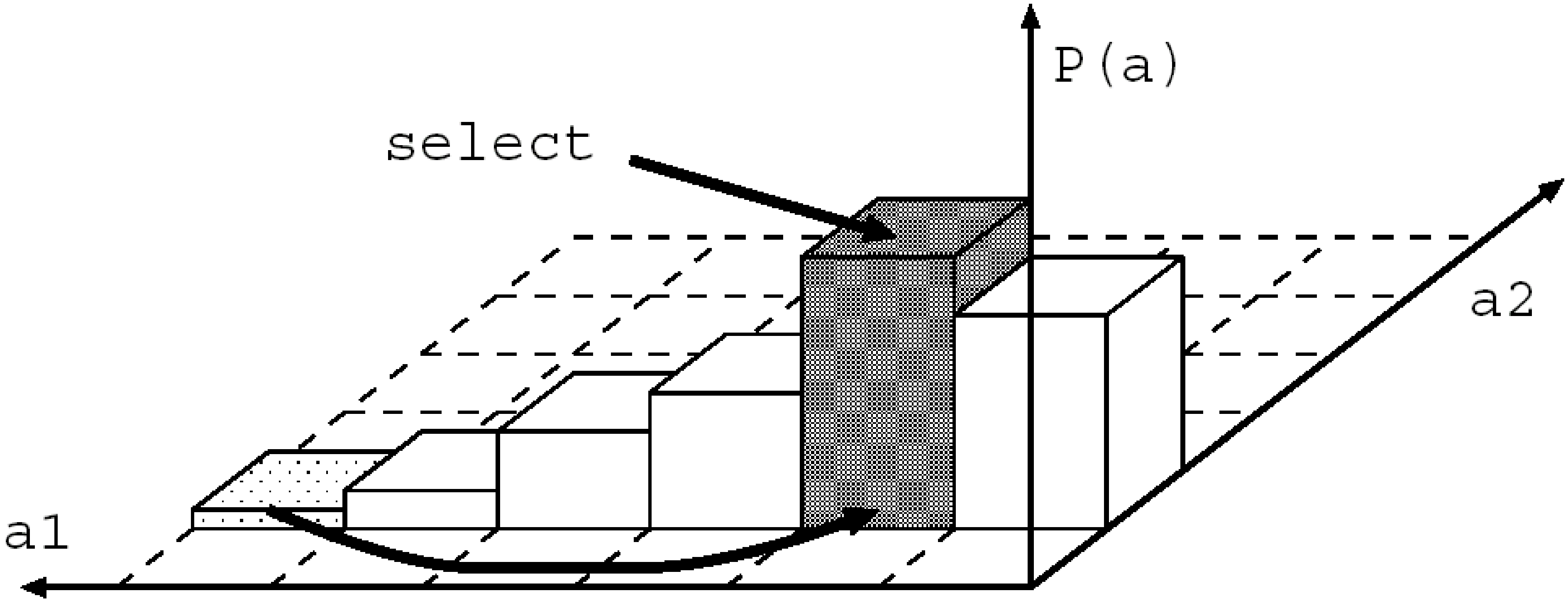


先ほどのボルツマン選択を Gibbsサンプリングしてみる

Q-learning

連続空間の関数近似

ボルツマン行動選択

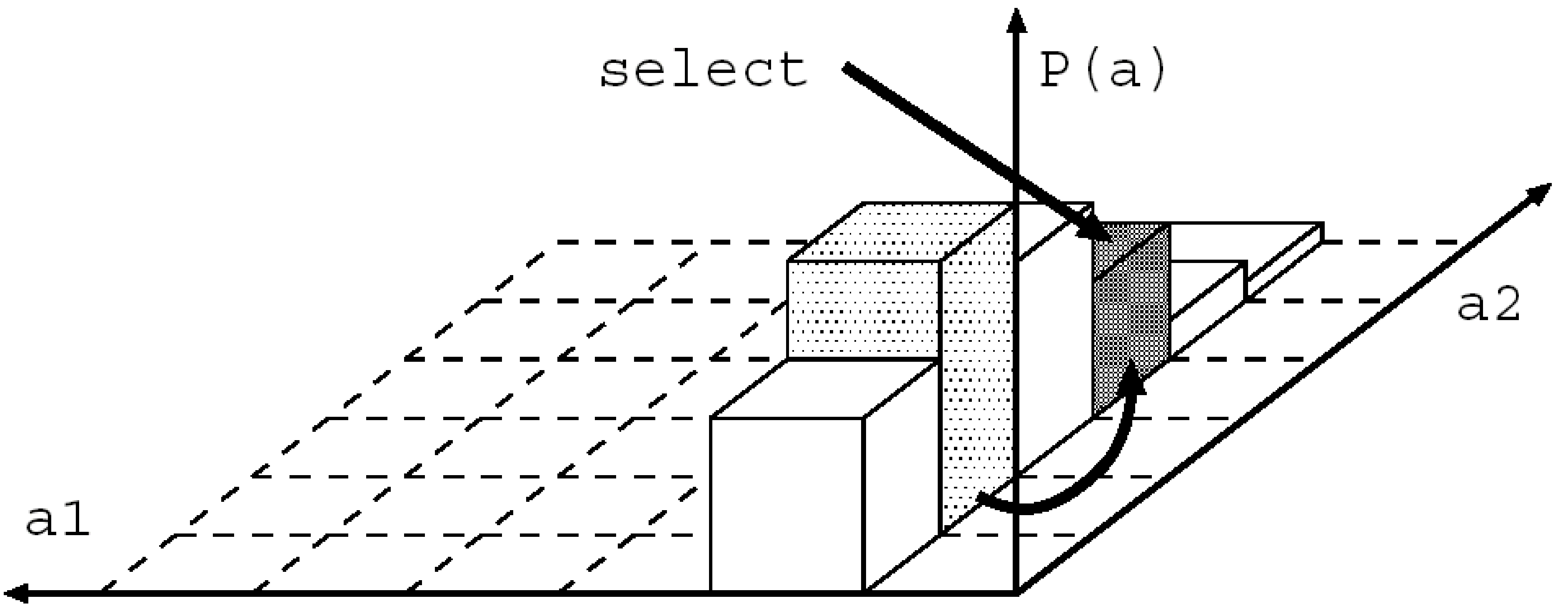


a2軸を固定し、a1軸についてのみボルツマン選択

Q-learning

連続空間の関数近似

ボルツマン行動選択

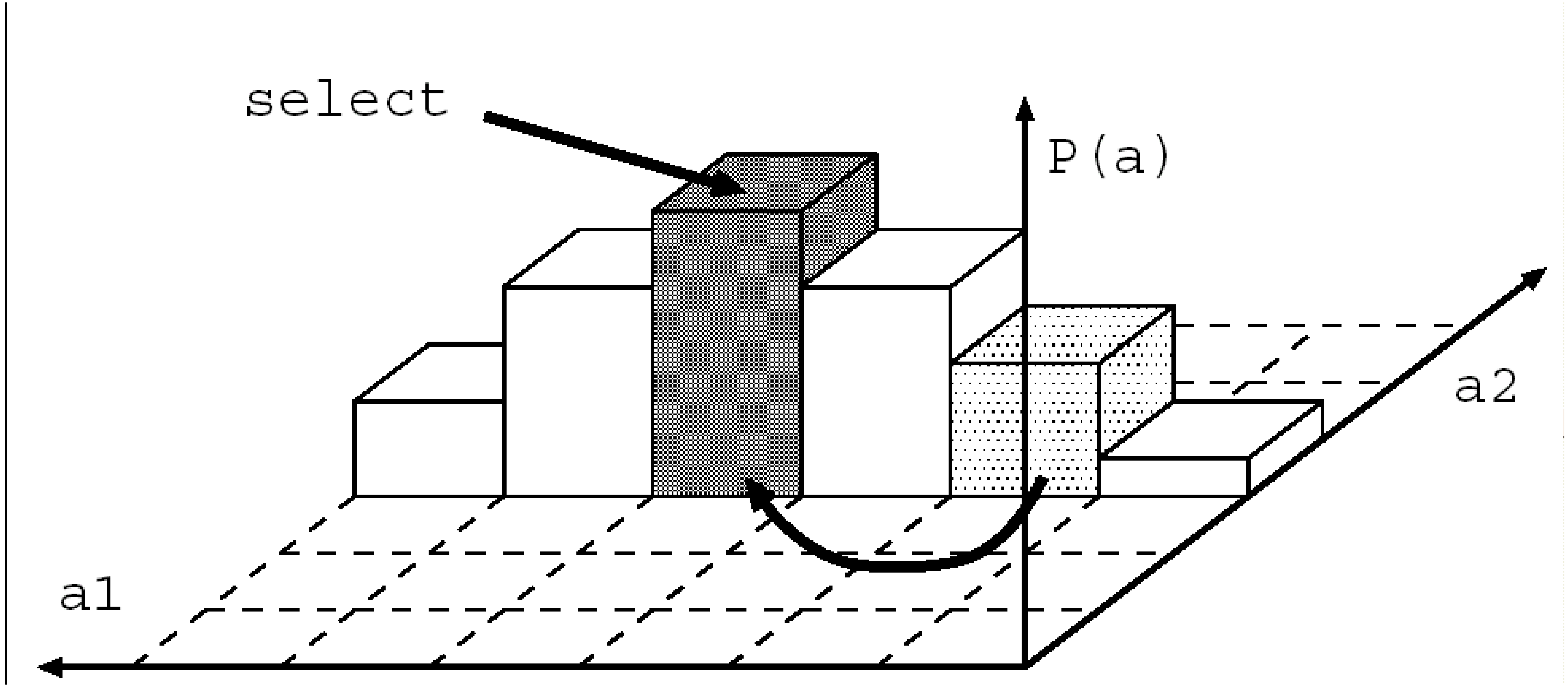


a1軸を先に選択された値に固定し、
今度はa2軸についてのみボルツマン選択

Q-learning

連続空間の関数近似

ボルツマン行動選択

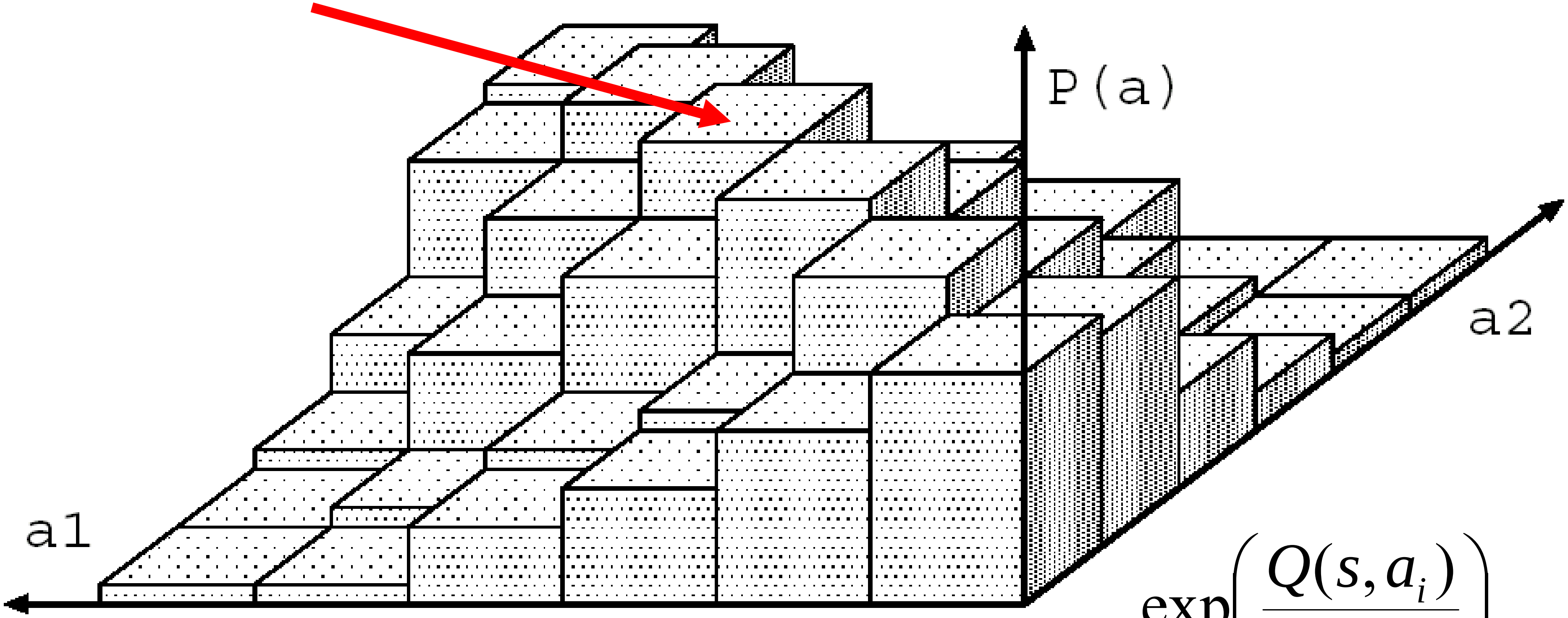


a2軸を先に選択された値に固定し、
再びa1軸についてのみボルツマン選択　これを繰返す

Q-learning

連続空間の
関数近似

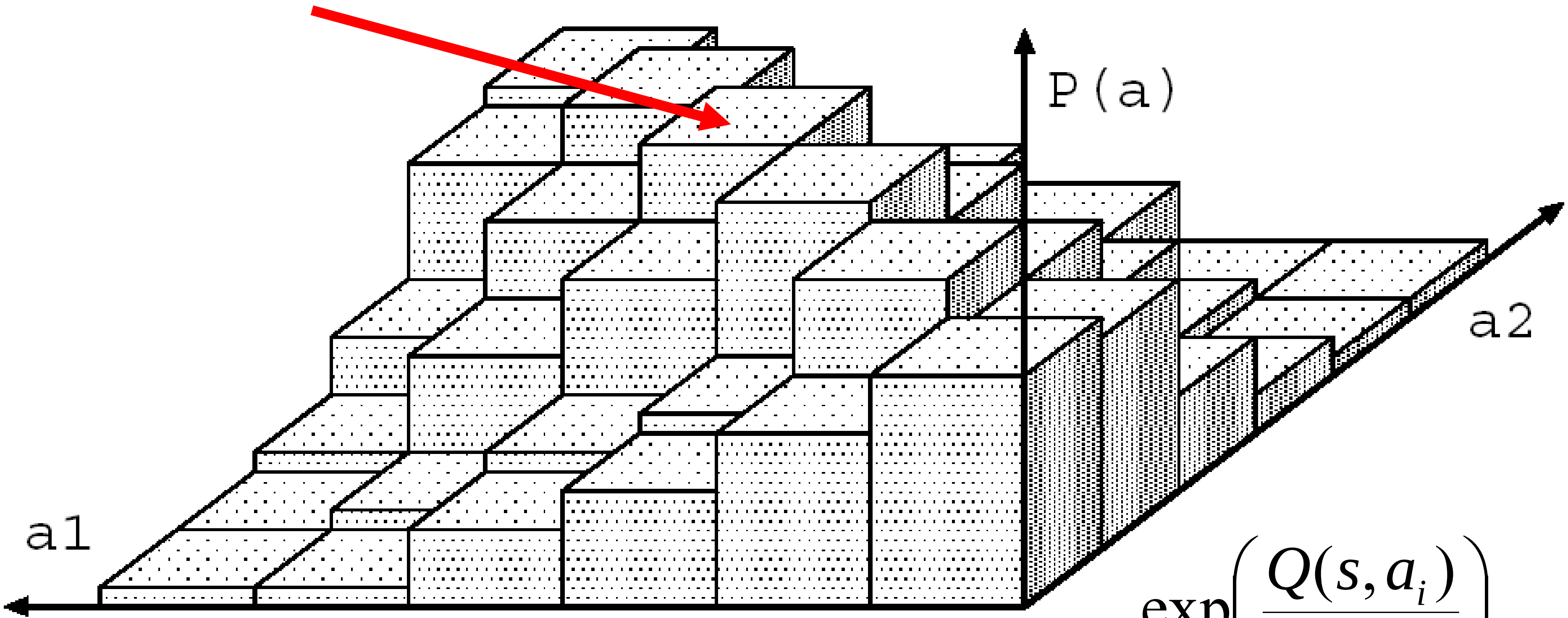
ボルツマン
行動選択



$$P(a_i) = \frac{\exp\left(\dfrac{Q(s,a_i)}{T}\right)}{\sum_{j=1}^{A} \exp\left(\dfrac{Q(s,a_j)}{T}\right)}$$

Gibbsサンプリング：
全行動空間の exp(Q) の合計を
計算せずに右式に従うサンプルを得る！

Q-learning	連続空間の関数近似	**ボルツマン行動選択**



$$P(a_i) = \frac{\exp\left(\dfrac{Q(s,a_i)}{T}\right)}{\sum_{j=1}^{A} \exp\left(\dfrac{Q(s,a_j)}{T}\right)}$$

Gibbsサンプリング：

全行動空間の exp(Q) の合計を
計算せずに右式に従うサンプルを得る！

Q-learning

連続空間の関数近似

ボルツマン行動選択

Gibbsサンプリングによる行動選択処理

ただし、
MaxQ を求める
計算が「近似」に

SARSAアルゴリズムなら問題ない

特長

・実装が簡単
・空間爆発を回避
・足りなければ計算反復の回数を増やすだけ
・分割を細かくしても、計算はあまり変わらない

Q-learning

連続空間の
関数近似

ボルツマン
行動選択

Gibbsサンプリングによる
行動選択処理

ただし、
MaxQ を求める
計算が「近似」に

SARSAアルゴリズムなら問題ない

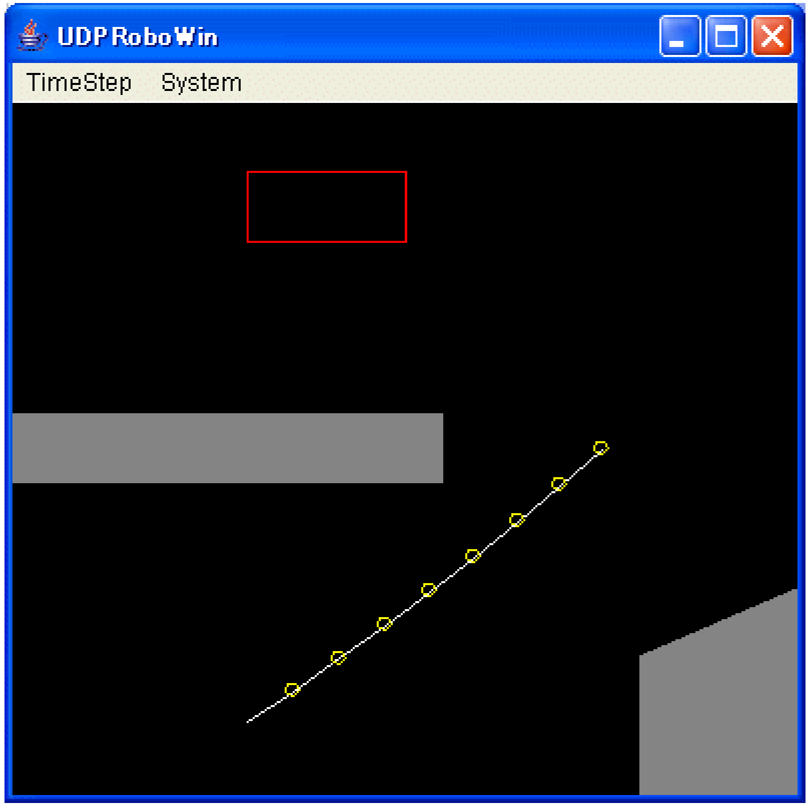
特長

ランダム矩形タイリング
による関数近似法との
計算処理の相性が良い

- 実装が簡単
- 空間爆発を回避
- 足りなければ計算反復の
  回数を増やすだけ
- 分割を細かくしても、計算
  はあまり変わらない

シミュレーション： Multi-Joint Arm

状態8次元
行動8次元



特徴ベクトルの設定

1）ランダムタイリング
8次元空間を10^8 の格子状とし、
タイルの境界をこの格子に合わせた
ランダムな大きさのタイルを200個生成
各次元要素をタイルに選択する確率=0.3

2）等間隔グリッドタイリング
8次元空間を 2^8 の等間隔グリッドで
分割し、2^8=256個のタイル生成

Q-learning の設定

割引率 γ=0.9
温　度　T=0.4
学習率 α=0.5

Gibbs-Sampling の設定

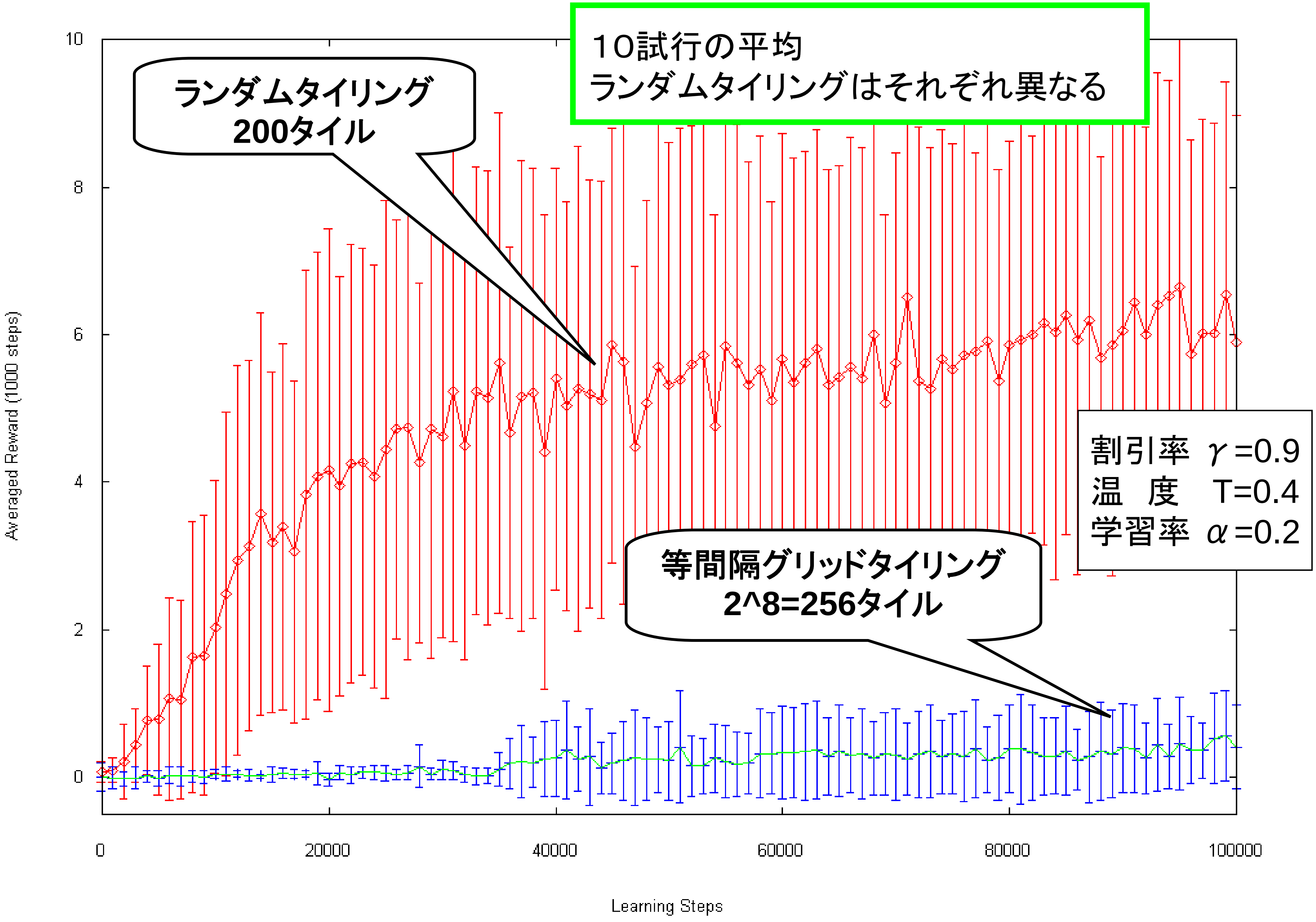
反復回数：40回×8次元

タイル修正のための
データエピソード：

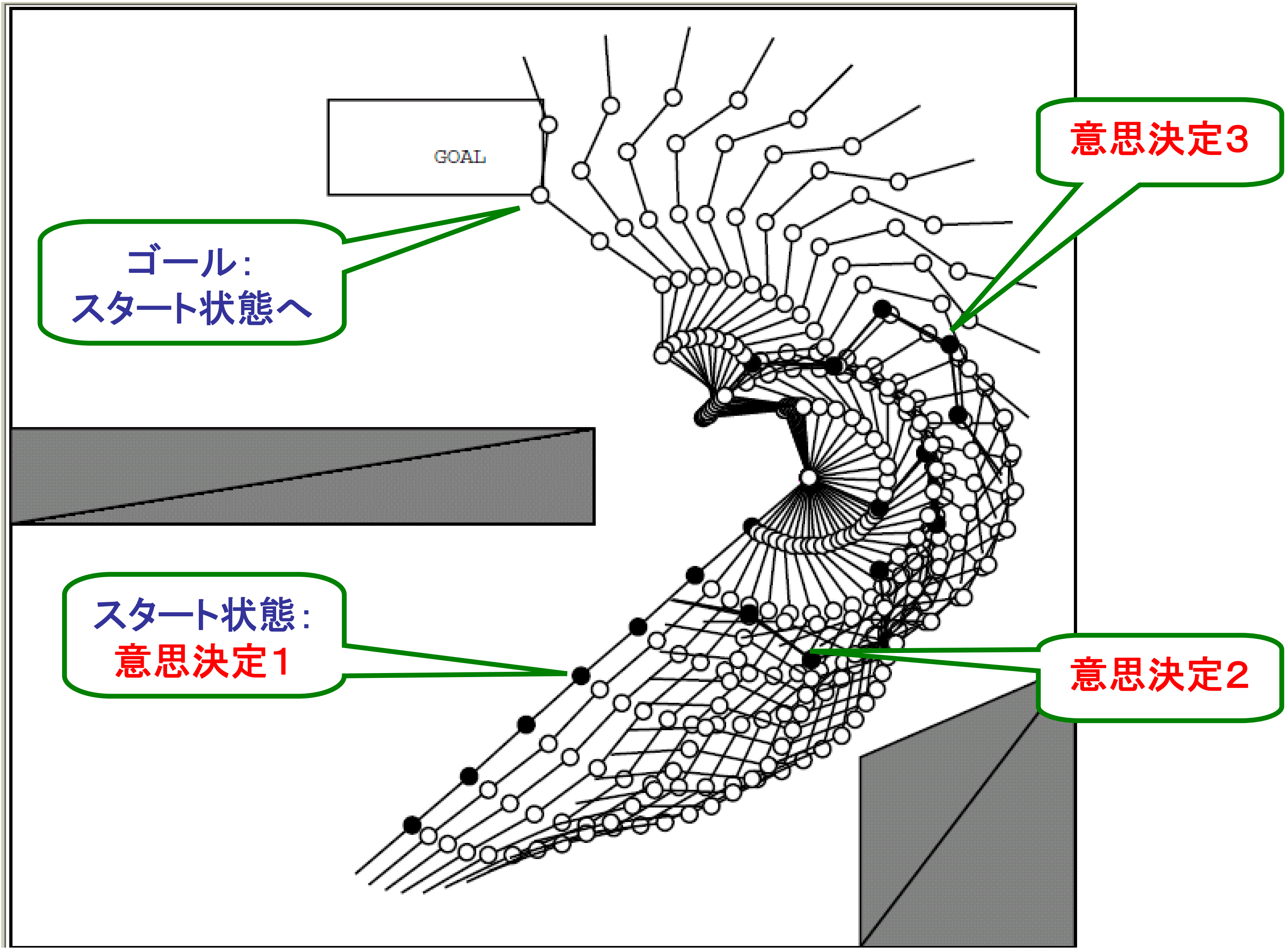
学習初期 2000 step

シミュレーション結果： Multi-Joint Arm

状態8次元
行動8次元



学習で得られた動作の一例



まとめ

「ランダム矩形タイルによる汎化方法と
Gibbsサンプリングによる行動選択方法」

Q-learning

連続空間の関数近似

ボルツマン行動選択

MaxQ を求める
計算が近似に

ランダムタイリングによる高次元連続空間の汎化

Gibbsサンプリングによる行動選択処理

相性が良い

- **実装が簡単**
- **空間爆発を回避**
- **足りなければタイルを増やすだけ**
- **実装が問題に依存しない**

- **実装が簡単**
- **空間爆発を回避**
- **足りなければ計算反復の回数を増やすだけ**
- **分割を細かくしても、計算はあまり変わらない**

高次元連続状態-行動空間の強化学習問題へ実装