Learning Heuristic Functions Through Supervised Learning

Master Thesis

Đorđe Relić

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Artificial Intelligence

June 30, 2017

Outline

1. Background
   - MDP
2. Learning a Heuristic Function
   - Machine Learning
   - Gradient Descent Methods
3. Results
   - Experimental environment
   - Parameters
   - Final result
4. Conclusion and Future Work

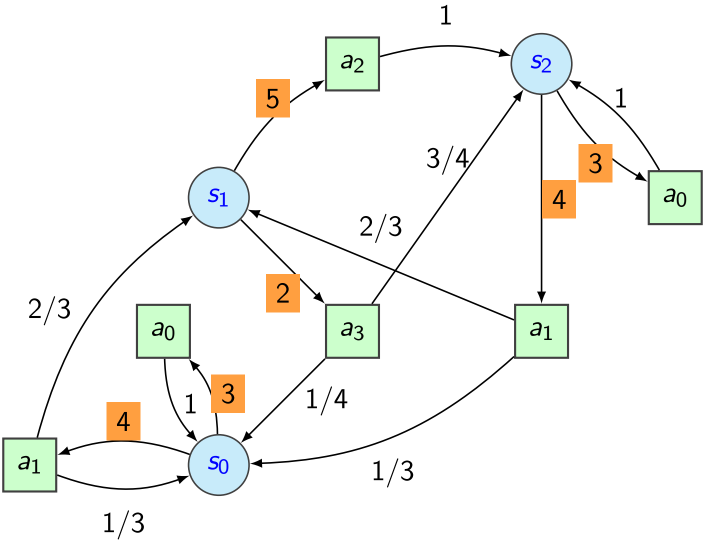
MDP

Definition (Markov Decision Process)

A MDP is a 6-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, H, s_0 \rangle$ where:

- $\mathcal{S}$ – the finite set of states
- $\mathcal{A}$ – the finite set of actions
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ – the **transition function**
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ – the reward function
- $H \in \mathbb{N}$ – the **finite horizon**
- $s_0 \in \mathcal{S}$ – the **initial state**

Example of a MDP



MDP

A policy $\pi : \mathcal{S} \times \{1, ..., H\} \rightarrow \mathcal{A}$.
State-values: $V_\pi(s, d)$ and action value: $Q_\pi(s, \pi(s, d))$.

MDP

A policy $\pi : \mathcal{S} \times \{1, ..., H\} \rightarrow \mathcal{A}$.
State-values: $V_\pi(s, d)$ and action value: $Q_\pi(s, \pi(s, d))$.
A factored MDP:

- A finite-domain variable v associated with $\mathcal{D}_v$
- A finite set of finite-domain variables $\mathcal{V}$
- Define state using $\mathcal{V}$
- For state $s \in \mathcal{S}$ we have a set of facts $\mathcal{F}(s) = \{\langle v, d\rangle \mid v \in \mathcal{V} \wedge d \in \mathcal{D}_v\}$

MDP

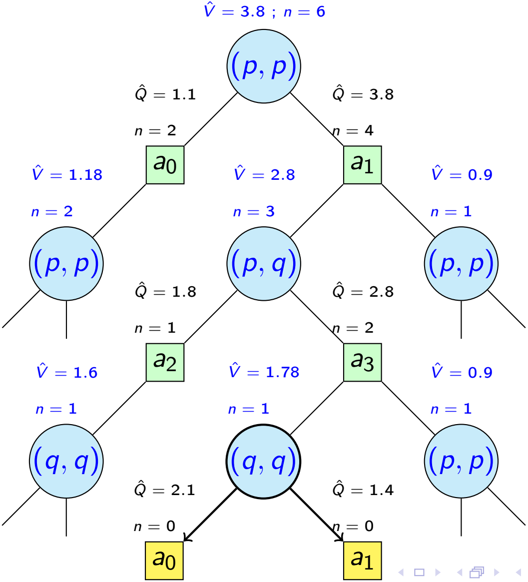
A policy $\pi : \mathcal{S} \times \{1, ..., H\} \to \mathcal{A}$.
State-values: $V_\pi(s, d)$ and action value: $Q_\pi(s, \pi(s, d))$.
A factored MDP:

- A finite-domain variable v associated with $\mathcal{D}_v$
- A finite set of finite-domain variables $\mathcal{V}$
- Define state using $\mathcal{V}$
- For state $s \in \mathcal{S}$ we have a set of facts $\mathcal{F}(s) = \{\langle v, d\rangle \mid v \in \mathcal{V} \wedge d \in \mathcal{D}_v\}$

Search algorithm: UCT*

- Action Selection
- Expansion
- Backup

UCT* - Expansion



Outline

1. Background
   - MDP
2. Learning a Heuristic Function
   - Machine Learning
   - Gradient Descent Methods
3. Results
   - Experimental environment
   - Parameters
   - Final result
4. Conclusion and Future Work

Learning a Heuristic Function

Machine Learning

Automated learning from data.

Learning a Heuristic Function

Machine Learning

Automated learning from data.

For each action $a \in \mathcal{A}$:

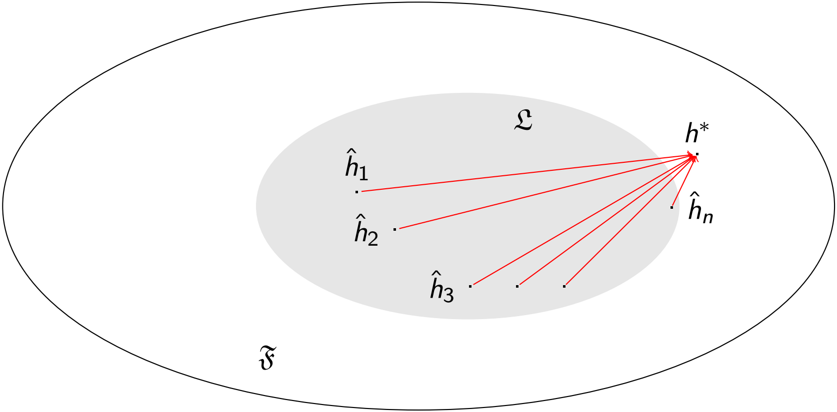
- We want to learn $h^* : \mathcal{S} \to \mathbb{R}$
- A given data set $\mathfrak{D} = \{(s, \hat{Q})\}_{i=1}^{m}$ with samples $h^*(s) = \hat{Q}$
- We want to find $\hat{h} : \mathcal{S} \to \mathbb{R}$ which approximates h^*
- Search in $\mathfrak{L}$ – the space of linear functions

$$\hat{h}(s) = \sum_{f \in \mathcal{F}(s)} w_f$$

- Evaluating $\hat{h}$ with the **Mean Square Error** function:

$$J(\mathcal{W}, s) = \sum_{(s,\hat{Q}) \in \mathcal{D}} (\hat{h}(s) - \hat{Q})^2$$

Learning a Heuristic Function



Gradient Descent Methods

Update of weights in the opposite direction of the MSE gradient.

$$w_f := w_f - \alpha \frac{\partial J(\mathcal{W}, s)}{\partial w_f}, \forall w_f \in \mathcal{W}$$

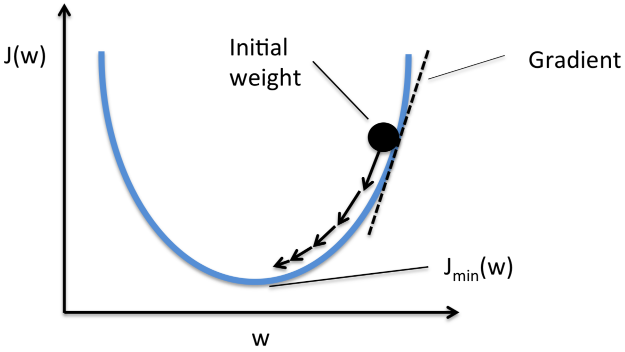
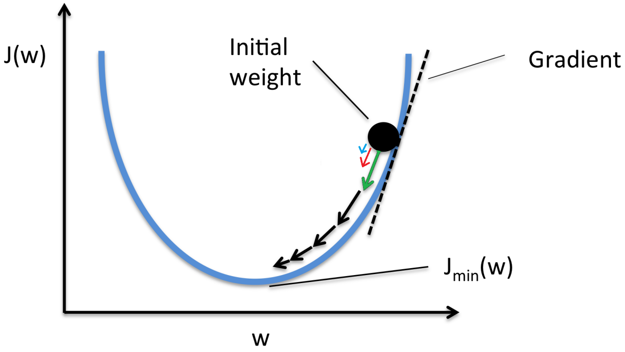


Image credits: Sebastian Raschka (https://sebastianraschka.com).

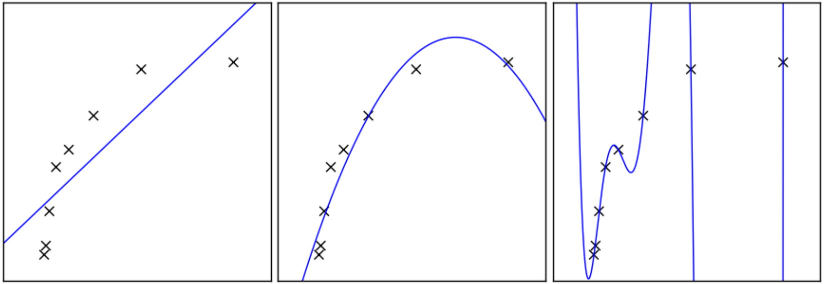
Gradient Descent Methods

Based on the portion of $\mathfrak{D}$ we have three gradient descent types:

- Stochastic Gradient Descent – each data set entry
- Batch Gradient Descent – whole data set
- Mini Batch Gradient Descent – subset of the data set



Underfitting and Overfitting



d = 1 (under-fit) · d = 2 · d = 6 (over-fit)

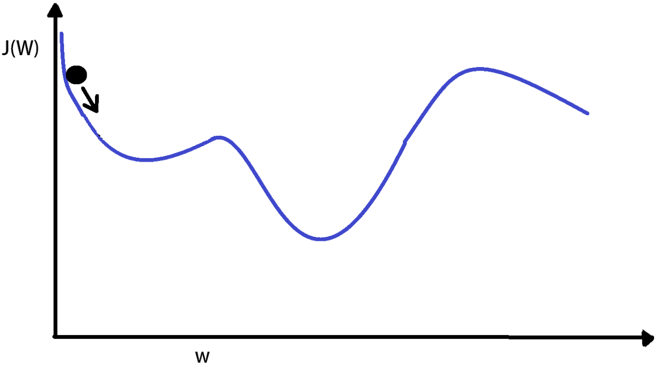
Image credits: Radim Rehurek (http://radimrehurek.com).

Gradient Descent Methods

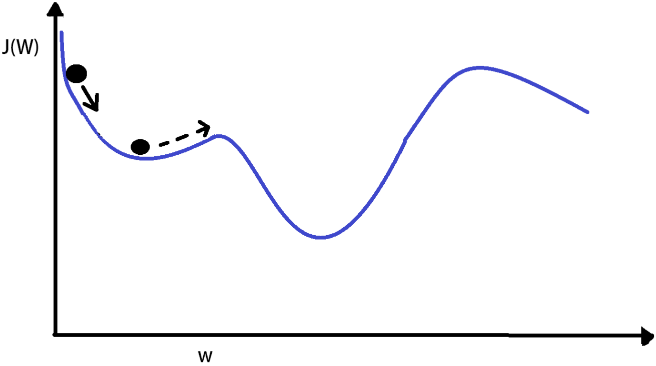
Improvements:

- Early stopping
- Learning rate decay
- Momentum

Momentum



Momentum



Gradient Descent Methods

To recap what we have:

- A data set $\mathfrak{D} = \{(s, \hat{Q})\}_{i=1}^{n}$
- Function approximation:

$$\hat{h}(s) = \sum_{f \in \mathcal{F}(s)} w_f$$

- Parameters – GD type, number of epochs, learning rate, momentum, learning rate decay, early stopping

Outline

1. Background
   - MDP
2. Learning a Heuristic Function
   - Machine Learning
   - Gradient Descent Methods
3. Results
   - Experimental environment
   - Parameters
   - Final result
4. Conclusion and Future Work

Results

Experimental environment:

- 12 domains and 10 instances per domain from IPPC 2014 and IPPC 2011
- Benchmarking against
  - DP-UCT
  - Winning algorithm of IPPC 2011
  - Winning algorithm of IPPC 2014

Goal

Find a parameter configuration for learning a heurstic function which outperforms baseline algorithms.

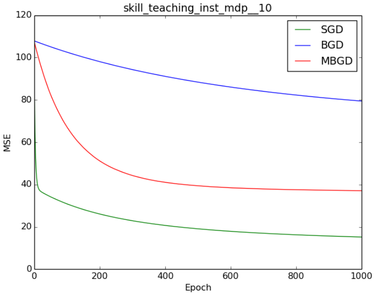
Results

Parameters:

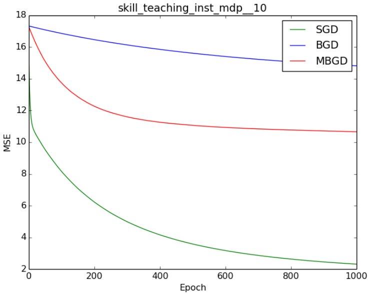
- Heuristic Function
- Data set size
- Gradient Descent Types
- Learning rate
- Number of epochs
- Momentum and learning rate decay
- Combination of features

Heuristic Functions

We wanted to learn two heuristic functions: IDS and UCT* using IDS.



a. IDS



b. UCT* using IDS

Figure: $\alpha = 0.00001$, $\sigma = 0.05$, $\gamma = 0$ and number of epochs 1000

Results

Parameters:

- Heuristic Function – **IDS** $\sim$ 75 and **UCT*** $\sim$ 35
- Data set size
- Gradient Descent Types
- Learning rate
- Number of epochs
- Momentum and learning rate decay
- Combination of features

Results

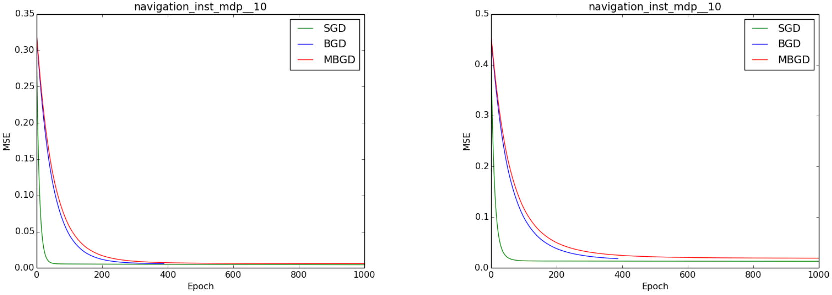
Parameters:

- Heuristic Function – **IDS** ~ 75 and **UCT*** ~ 35
- Data set size – **Bigger data set is better**
- Gradient Descent Types
- Learning rate
- Number of epochs
- Momentum and learning rate decay
- Combination of features

Gradient Descent Types

SGD performing best:

DP-UCT	IPPC2011	IPPC2014	SGD	BGD	MBGD
0.0	0.0	0.75	**1.0**	0.63	0.81



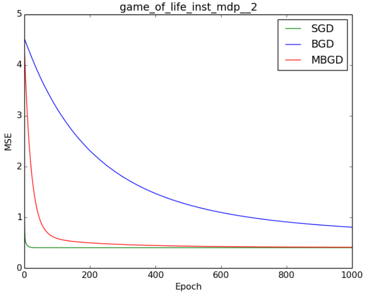
a. Training b. Testing

Figure: $\alpha = 0.0001$, $\sigma = 0$, $\gamma = 0$ and number of epochs 1000

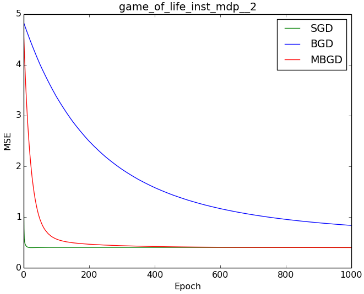
Gradient Descent Types

BGD performing best:

DP-UCT	IPPC2011	IPPC2014	SGD	BGD	MBGD
0.71	0.93	0.75	0.91	**1.0**	0.85



a. Training



b. Testing

Figure: $\alpha = 0.0005$, $\sigma = 0.05$, $\gamma = 0$ and number of epochs 1000

Results

Parameters:

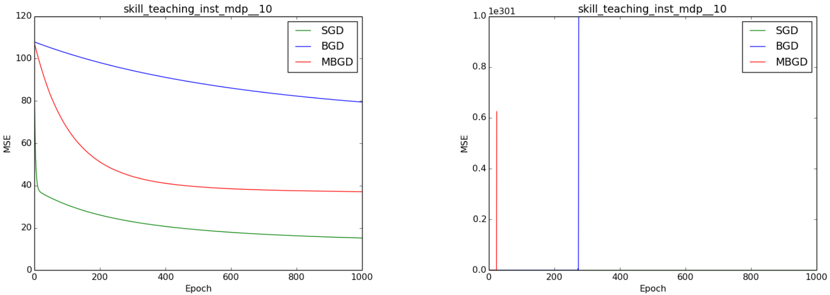
- Heuristic Function – **IDS** $\sim$ 75 and **UCT*** $\sim$ 35
- Data set size – **Bigger data set is better**
- Gradient Descent Types – **Best choice depends on the task**
- Learning rate
- Number of epochs
- Momentum and learning rate decay
- Combination of features

Results

Parameters:

- Heuristic Function – **IDS** $\sim$ 75 and **UCT*** $\sim$ 35
- Data set size – **Bigger data set is better**
- Gradient Descent Types – **Best choice depends on the task**
- Learning rate – **Best choice depends on the task**
- Number of epochs – **Best choice depends on the task**
- Momentum and learning rate decay
- Combination of features

Momentum



a. $\gamma = 0$ b. $\gamma = 0.3$

Figure: IDS, $\alpha = 0.00001$, $\sigma = 0.05$, $\gamma = 0$ and number of epochs 1000

Results

Parameters:

- Heuristic Function – **IDS** $\sim$ 75 and **UCT*** $\sim$ 35
- Data set size – **Bigger data set is better**
- Gradient Descent Types – **Best choice depends on the task**
- Learning rate – **Best choice depends on the task**
- Number of epochs – **Best choice depends on the task**
- Momentum and learning rate decay – **No momentum and small learning rate decay**
- Combination of features

Combination of features

Combining state features into pairs and generating new features.

$$\mathcal{F}^2(s) =$$
$$\{\langle v_1, d_1, v_2, d_2\rangle \mid v_1, v_2 \in \mathcal{V} \wedge d_1, d_2 \in \mathcal{D}_v \wedge s[v_1] = d_1 \wedge s[v_2] = d2\}$$

Combination of features

Combining state features into pairs and generating new features.

$$\mathcal{F}^2(s) =$$
$$\{\langle v_1, d_1, v_2, d_2\rangle \mid v_1, v_2 \in \mathcal{V} \wedge d_1, d_2 \in \mathcal{D}_v \wedge s[v_1] = d_1 \wedge s[v_2] = d2\}$$

Outcome:

- Number of new features $\binom{n}{2} - \frac{n}{2}$
- More accurate action-values but worse result
- More exploitation
- Less trials

Results

Parameters:

- Heuristic Function – **IDS** $\sim$ 75 and **UCT*** $\sim$ 35
- Data set size – **Bigger data set is better**
- Gradient Descent Types – **Best choice depends on the task**
- Learning rate – **Best choice depends on the task**
- Number of epochs – **Best choice depends on the task**
- Momentum and learning rate decay – **No momentum and small learning rate decay**
- Combination of features – **Without expected improvement**

Results

Final result

No single parameter configuration which outperforms the baseline algorithms.

Results

Final result

No single parameter configuration which outperforms the baseline algorithms.

There is a *Set of Best Parameter Configurations* for which the baseline algorithms are outperformed.

Results

	DP-UCT	IPPC 2011	IPPC 2014	SBPC OH
Wildfire	0.79	0.83	0.69	**0.86**
Triangle	0.42	0.39	**0.88**	0.7
Academic	**0.68**	0.3	0.31	0.6
Elevators	0.59	**0.96**	**0.96**	0.9
Tamarisk	0.64	**0.9**	0.88	**0.9**
Sysadmin	0.61	0.74	0.8	**0.99**
Recon	0.58	**0.98**	0.94	**0.98**
Game	0.71	0.91	**0.97**	0.96
Traffic	0.86	0.93	**0.98**	0.7
Crossing	0.42	0.81	**0.99**	0.83
Skill	0.93	0.93	0.94	**1.0**
Navigation	0.67	0.58	0.92	**1.0**
Total	0.66	0.77	0.86	**0.87**

Conclusion

"All models are wrong, but some models are useful." - George Box

Conclusion

"All models are wrong, but some models are useful." - George Box

Conclusion:

- No single parameter configuration for learning an offline heuristic in domain independent probabilistic planning.
- Many parameters which demand a large number of experiments and fine tuning.
- A set of parameter configurations was found that outperforms the baseline algorithms for 0.01.

Future Work

Future work:

- Other Gradient descent methods
- More domain knowledge for better analysis of the data set
- Investigating the underperformance of combined features approach
- Multiple iterations of learning
- Online learning

Questions?

Additional slides

Policy

Definition (Value functions)

Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, H, s_0 \rangle$ be a MDP, $s \in \mathcal{S}$ a state and π a policy. The **state-value** $V_\pi(s, d)$ of s under π with d steps to go is defined as

$$V_\pi(s, d) = Q_\pi(s, \pi(s, d))$$

where the **action-value** $Q_\pi(s, \pi(s, d))$ under π is defined as

$$Q_\pi(s, a) = \begin{cases} \mathcal{R}(s, a) + \sum_{s' \in \mathcal{S}} (\mathbb{P}_\mathcal{T}[s' | s, a] \cdot V_\pi(s', d - 1)) & , d > 0 \\ \mathcal{R}(s, a) & , d = 0 \end{cases}$$

for all state-action pairs (s, a).

Policy

Definition (Optimal policy)

Let the Bellman optimality equation for a state $s \in \mathcal{S}$ be a set of equations that describe $V^*(s, d)$, where

$$V^*(s, d) = \max_{a \in \mathcal{A}} Q^*(s, d, a)$$

$$Q^*(s, d, a) = \begin{cases} \mathcal{R}(s, a) + \sum_{s' \in \mathcal{S}} (\mathbb{P}_{\mathcal{T}}[s' | s, a] \cdot V^*(s', d - 1)) & , d > 0 \\ \mathcal{R}(s, a) & , d = 0 \end{cases}$$

A policy π^* is an **optimal policy** if $\pi^* \in \arg\max_{a \in \mathcal{A}} Q^*(s, d, a)$ for all $s \in \mathcal{S}$.

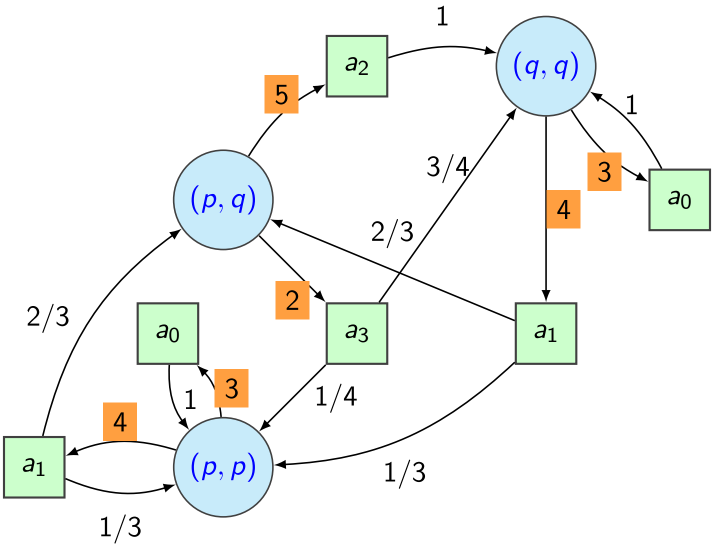
Planning task

Definition (Planning task)

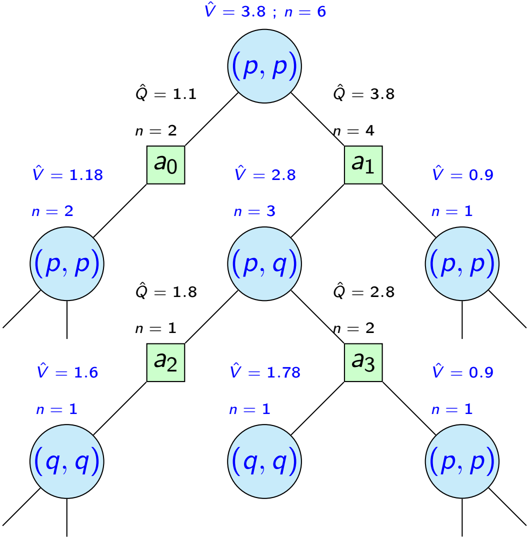
A **planning task** is a 4-tuple $T = \langle \mathcal{V}, \mathcal{A}, H, s_0 \rangle$ where:

- $\mathcal{V}$ – the finite set **of finite-domain variables** v with domain $\mathcal{D}_v$
- $\mathcal{A}$ – the finite set of **actions** $\langle effect_a, reward_a \rangle = a \in \mathcal{A}$ where
  - $effect_a$ – is a probability distribution over partial variable assignment $\{(p_i^a, e_i^a)\}_{i=1}^n$ where p_i^a is a probability, e_i^a is a partial variable assignment and $\sum_{i=1}^{n} p_i^a = 1$
  - $reward_a$ – the reward of applying action a
- $H \in \mathbb{N}$ – the **finite horizon**
- $s_0 \in \mathcal{V}$ – the **initial state**

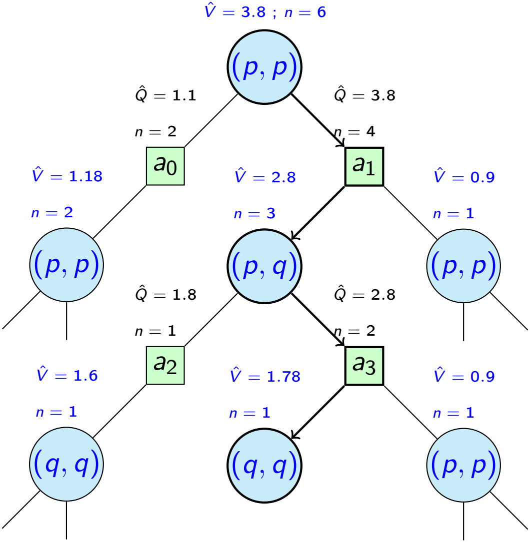
Example of a factored MDP



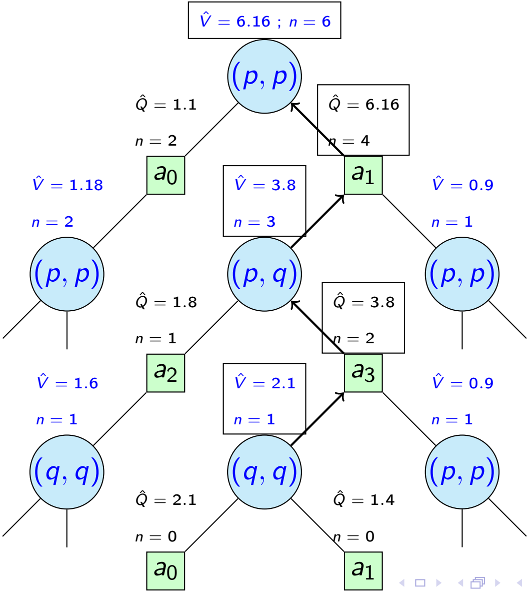
UCT* - Action Selection



UCT* - Action Selection



UCT* - Backup


$\hat{V} = 6.16$; $n = 6$
(p, p)
$\hat{Q} = 1.1$
$n = 2$
$\hat{Q} = 6.16$
$n = 4$
a_0
a_1
$\hat{V} = 1.18$
$n = 2$
$\hat{V} = 3.8$
$n = 3$
$\hat{V} = 0.9$
$n = 1$
(p, p)
(p, q)
(p, p)
$\hat{Q} = 1.8$
$n = 1$
$\hat{Q} = 3.8$
$n = 2$
a_2
a_3
$\hat{V} = 1.6$
$n = 1$
$\hat{V} = 2.1$
$n = 1$
$\hat{V} = 0.9$
$n = 1$
(q, q)
(q, q)
(p, p)
$\hat{Q} = 2.1$
$n = 0$
$\hat{Q} = 1.4$
$n = 0$
a_0
a_1

Gradient Descent Methods

Gradient descent methods:

$$w_f := w_f - \alpha \frac{\partial J(\mathcal{W}, s)}{\partial w_f}, \forall w_f \in \mathcal{W}$$

Where we have the error function over the data set $\mathfrak{D}$':

$$J(\mathcal{W}, s) = \frac{1}{2m} \sum_{(s,\hat{Q}) \in \mathfrak{D}} (\hat{h}(s) - \hat{Q})^2$$

and by deriving we get:

$$\frac{\partial}{\partial w_f} J(\mathcal{W}, s) = \frac{1}{m} \sum_{(s,\hat{Q}) \in \mathfrak{D}} (\hat{h}(s) - \hat{Q}) \cdot \frac{\partial}{\partial w_f} \hat{h}(s), \forall w_f \in \mathcal{W}$$

Gradient Descent Methods

Based on the portion of $\mathfrak{D}$ we have three gradient descent types:

Gradient Descent Methods

Based on the portion of $\mathfrak{D}$ we have three gradient descent types:

- Stochastic Gradient Descent

$$\frac{\partial}{\partial w_f} J(\mathcal{W}, s) = (\hat{h}(s) - \hat{Q}) \cdot \frac{\partial}{\partial w_f} \hat{h}(s), \forall w_f \in \mathcal{W}$$

- Batch Gradient Descent

$$\frac{\partial}{\partial w_f} J(\mathcal{W}, s) = \frac{1}{m} \sum_{(s,\hat{Q}) \in \mathfrak{D}} (\hat{h}(s) - \hat{Q}) \cdot \frac{\partial}{\partial w_f} \hat{h}(s), \forall w_f \in \mathcal{W}$$

- Mini Batch Gradient Descent

$$\frac{\partial}{\partial w_f} J(\mathcal{W}, s) = \frac{1}{k - j} \sum_{(s,\hat{Q}) \in \mathfrak{B}} (\hat{h}(s) - \hat{Q}) \cdot \frac{\partial}{\partial w_f} \hat{h}(s), \forall w_f \in \mathcal{W}$$