Temporal Difference Learning and TDMC

Seminar aus maschinellem Lernen

Jannik Abbenseth

December 2, 2014

Agenda

Learning Objective

- What to achieve?
  - Evaluation function for all states

$$V(\mathbf{x}_t, \mathbf{w}) := \mathbf{x}_t^{\mathsf{T}} \cdot \mathbf{w}.$$

    - ★ $\mathbf{x}_t = (x_{t1}, x_{t2}, \ldots, x_{tM})^T$: Feature vector.
    - ★ $\mathbf{w} = (w_1, w_2, \ldots, w_M)^T$: Weights of features.

Learning Method

- Update of Evaluation function:

$$V(\mathbf{x}_t, \mathbf{w}) := V(\mathbf{x}_t, \mathbf{w}) + \alpha \Delta V(\mathbf{x}_t, \mathbf{w}).$$

  Where α controls the learning rate.
- Because $\mathbf{x}_t$ is fixed this can be rewritten to:

$$\mathbf{w} := \mathbf{w} + \alpha \Delta \mathbf{w}.$$

- $\mathbf{w}$ only gets updated after an observation.

TD(0)

- With a state P_{t+1} selected by an ϵ-greedy Policy:

$$\Delta V(\mathbf{x_t}, \mathbf{w}) = r_t + V(\mathbf{x}_{t+1}, \mathbf{w}) - V(\mathbf{x}_t, \mathbf{w})$$

  where r_t is a immediate reward from the executed action.
- The new weight vector $\mathbf{w}$ after one observation:

$$\mathbf{w} = \mathbf{w} + \alpha \sum_t \Delta V(\mathbf{x}_t, \mathbf{w})\mathbf{x}_t$$

- TD(0) only takes the immediate successor into account for learning.
- TD(0) relies strongly on the evaluation function V.

TD(1)

- Considers all successors.
- Learns from real outcome.
- Error gets the sum of changes in the predictions.

$$z - V(\mathbf{x}_t, \mathbf{w}) = \sum_{k=t}^{T-1} r_k + V(\mathbf{x}_{k+1}, \mathbf{w}) - V(\mathbf{x}_k, \mathbf{w}),$$

with $V(\mathbf{x}_T, \mathbf{w}) = z$, where z is the outcome for the sequence.

TD(1)

- So $\Delta\mathbf{w_t}$ gets

$$\Delta\mathbf{w}_t = \sum_{k=t}^{T-1} (r_k + V(\mathbf{x}_{k+1}, \mathbf{w}) - V(\mathbf{x}_k, \mathbf{w}))\nabla_w V(\mathbf{x}_k, \mathbf{w}).$$

- And $\mathbf{w}$ gets

$$\mathbf{w} := \mathbf{w} + \alpha \sum_t (r_t + V(\mathbf{x}_{t+1}, \mathbf{w}) - V(\mathbf{x}_t, \mathbf{w})) \sum_{k=1}^{t} \nabla_w V(\mathbf{x}_k, \mathbf{w})$$

TD(λ)

- Introducing a discount factor $\lambda \in [0, 1]$ for $\Delta\mathbf{w}_t$ so that

$$\mathbf{w} := \mathbf{w} + \alpha \sum_t (r_t + V(\mathbf{x}_{t+1}, \mathbf{w}) - V(\mathbf{x}_t, \mathbf{w})) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w V(\mathbf{x}_k, \mathbf{w})$$

- λ determines the temporal span of influence.

Summary TD(λ)

- TD(0)

$$\mathbf{w} := \mathbf{w} + \alpha \sum_t (r_t + V(\mathbf{x}_{t+1}, \mathbf{w}) - V(\mathbf{x}_t, \mathbf{w})) \nabla_w V(\mathbf{x}_t, \mathbf{w})$$

- TD(1)

$$\mathbf{w} := \mathbf{w} + \alpha \sum_t (r_t + V(\mathbf{x}_{t+1}, \mathbf{w}) - V(\mathbf{x}_t, \mathbf{w})) \sum_{k=1}^{t} \nabla_w V(\mathbf{x}_k, \mathbf{w})$$

- TD(λ)

$$\mathbf{w} := \mathbf{w} + \alpha \sum_t (r_t + V(\mathbf{x}_{t+1}, \mathbf{w}) - V(\mathbf{x}_t, \mathbf{w})) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w V(\mathbf{x}_k, \mathbf{w})$$

Main Downside of TD(λ) and how to solve it

- The reward r_t exists only for terminal states and is assigned to all previous states.
  - Different rewards for a state if it exists in multiple games with different outcomes.
- Substitute the reward for non-terminal states.

Temporal Difference with Monte Carlo simulation

- *The goal is to maximize the total sum of substitute rewards provided by the environment.*
- Rewards are substituted by the probability of winning.
- Probability of winning is obtained by Monte Carlo Simulation:
  - Independent of previous moves.
  - Independent of evaluation function.
  - The closer to a terminal position, the closer to the actual outcome.

Return of P_t

- The Return R_t of a state P_t is given by:

$$R_t := \gamma^0 r_t + \gamma^1 r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{T-t} r_T = \sum_{i=t}^{T} \gamma^{i-t} r_i,$$

where r_t is the Monte Carlo simulation outcome for P_t and the discount factor $\gamma \in [0, 1]$.

n-step Return

- Additionally there is a n-step return, which uses the evaluation function in P_{t+n} for all subsequent substitute rewards:

$$R_t^{(n)} := \gamma^0 r_t + \gamma^1 r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(\mathbf{x}_{t+n}, \mathbf{w}).$$

λ-Return

- With a eligibility rate $\lambda \in [0,1]$ the target value for $V(\mathbf{x}_{t+n}, \mathbf{w})$ can be computed by:

$$\begin{aligned} R_t^\lambda :&= \lambda^0 R_t^{(1)} + \lambda^1 R_t^{(2)} + \ldots + \lambda^{T-t-2} R_t^{(T-t-1)} + \lambda^{T-t-1} R_t \\ &= \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t. \end{aligned}$$

Error function to optimize

- $V(\mathbf{x}_t, \mathbf{w})$ should be as close to R_t^λ as possible.
- Quadratic Error:

$$OF_{TDMC} = \sum_{t=1}^{T} \left(R_t^\lambda - V(\mathbf{x}_t, \mathbf{w}) \right)^2$$

Resulting update of **w**

- The gradient $\Delta \mathbf{w}$ can be written as:

$$\Delta \mathbf{w} := \frac{\delta OF_{TDMC}}{\delta \mathbf{w}} = -2 \sum_{t=1}^{T} \left[R_t^{\lambda} - V(\mathbf{x}_t, \mathbf{w}) \right] \nabla_{\mathbf{w}} V(\mathbf{x}_t, \mathbf{w})$$

with

$$\nabla_{\mathbf{w}} V(\mathbf{x_t}, \mathbf{w}) = \left(\frac{\delta V(\mathbf{x}_t, \mathbf{w})}{\delta \mathbf{w}_1}, \frac{\delta V(\mathbf{x}_t, \mathbf{w})}{\delta \mathbf{w}_2}, \ldots, \frac{\delta V(\mathbf{x}_t, \mathbf{w})}{\delta \mathbf{w}_M} \right)$$

Evaluation on Othello

- 15 Different features:
  - 10 represent a state on the board.
- Different game stages.

Training

- ϵ-greedy Policy with $\epsilon = 0.03$.
- Training with 1000 self-play games for TDMC(λ).
- Training with 5000 self-play games for TD(λ).

Results against untrained program

- TD(λ) as well as TDMC(λ) outperformed the untrained program.

TD(λ) vs. TDMC(λ)

- With d random moves for opening:

d	TDMC(λ) Player wins	Draws	TD(λ) Player wins
4	880	0	120
6	817	10	173
8	794	20	186
10	782	18	200

Discussion

- TDMC(λ) outperforms TD(λ) even for high ϵ.
  - TDMC(λ) bases learning on probability of winning in non-terminal positions.
  - Less dependent on game records.
  - Preferrable for self-play game learning.
- w is similar for end stage.
  - Monte Carlo simulation gives better approximation close to terminal positions.

References I

Richard S. Sutton.
Learning to Predict by the Methods of Temporal Differences.
Kluwer Academic Publishers, 1988.

Yasuhiro Osaki, Kazutomo Shibahara, Yasuhiro Tajima and Yoshiyuki Kotani
An Othello Evaluation Function Based on Temporal Difference Learning using Probability of Winning.
IEEE Symposium on Computational Intelligence and Games, 2008.

David Silver
Reinforcement Learning and Simulation-Based Search in Computer Go.