Reinforcement Learning

Lisbon Machine Learning School 2019

Stefan Riezler

Computational Lingustics & IWR
Heidelberg University, Germany
riezler@cl.uni-heidelberg.de

Overview

- Formalizing the reinforcement learning problem: **Markov Decision Processes** (MDPs)

Overview

- Formalizing the reinforcement learning problem: **Markov Decision Processes** (MDPs)
- **Dynamic programming** techniques for policy evaluation and policy optimization

Overview

- Formalizing the reinforcement learning problem: **Markov Decision Processes** (MDPs)
- **Dynamic programming** techniques for policy evaluation and policy optimization
- **Sampling-based techniques**: Monte-Carlo methods, Temporal-Difference learning, Q-learning

Overview

- Formalizing the reinforcement learning problem: **Markov Decision Processes** (MDPs)
- **Dynamic programming** techniques for policy evaluation and policy optimization
- **Sampling-based techniques**: Monte-Carlo methods, Temporal-Difference learning, Q-learning
- **Policy-gradient methods**: Score function gradient estimators, actor-critic methods

Overview

- Formalizing the reinforcement learning problem: **Markov Decision Processes** (MDPs)
- **Dynamic programming** techniques for policy evaluation and policy optimization
- **Sampling-based techniques**: Monte-Carlo methods, Temporal-Difference learning, Q-learning
- **Policy-gradient methods**: Score function gradient estimators, actor-critic methods
- **Seq2seq reinforcement learning**: Bandit structured prediction, actor-critic neural seq2seq learning

Overview

- Formalizing the reinforcement learning problem: **Markov Decision Processes** (MDPs)
- **Dynamic programming** techniques for policy evaluation and policy optimization
- **Sampling-based techniques**: Monte-Carlo methods, Temporal-Difference learning, Q-learning
- **Policy-gradient methods**: Score function gradient estimators, actor-critic methods
- **Seq2seq reinforcement learning**: Bandit structured prediction, actor-critic neural seq2seq learning
- **Off-policy/counterfactual** seq2seq reinforcement learning

Overview

- Formalizing the reinforcement learning problem: **Markov Decision Processes** (MDPs)
- **Dynamic programming** techniques for policy evaluation and policy optimization
- **Sampling-based techniques**: Monte-Carlo methods, Temporal-Difference learning, Q-learning
- **Policy-gradient methods**: Score function gradient estimators, actor-critic methods
- **Seq2seq reinforcement learning**: Bandit structured prediction, actor-critic neural seq2seq learning
- **Off-policy/counterfactual** seq2seq reinforcement learning
- **Seq2seq reinforcment learning** from **human feedback**

Textbooks

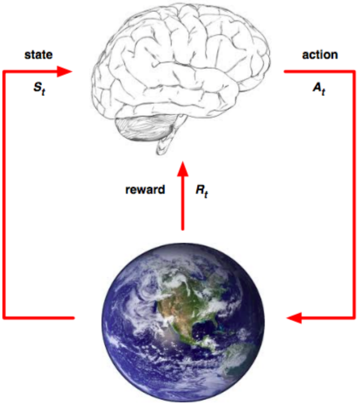
- Richard S. Sutton and Andrew G. Barto (2018, 2nd edition): Reinforcement Learning: An Introduction. MIT Press.
  - `http://incompleteideas.net/sutton/book/the-book-2nd.html`
- Csaba Szepesvári (2010). Algorithms for Reinforcement Learning. Morgan & Claypool.
  - `https://sites.ualberta.ca/~szepesva/RLBook.html`
- Dimitri Bertsekas and John Tsitsiklis (1996). Neuro-Dynamic Programming. Athena Scientific.
  - = another name for deep reinforcement learning, contains a lot of proofs, analog version can be ordered at `http://www.athenasc.com/ndpbook.html`

Reinforcement Learning (RL) Philosopy

- *Hedoninistic* learning system that *wants* something, and adapts its behavior in order to maximize a special signal or *reward* from its environment.
- *Interactive* learning by trial and error, using consequences of own actions in uncharted territory to learn to maximize expected reward.
- *Weak supervision signal* since no gold standard examples from expert are available.

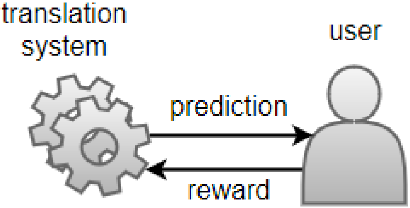
Reinforcement Learning Schema

- RL as Google DeepMind would like to see it (image from David Silver):



Reinforcement Learning Schema

- A real-world example: Interactive Machine Translation



  - action = predicting a target word
  - reward = per-sentence translation quality
  - state = source sentence and target history

Reinforcement Learning Schema

Agent/system and environment/user interact

- at each of a sequence of time steps $t = 0, 1, 2, \ldots$,
- where agent receives a state representation S_t,
- on which basis it selects an action A_t,
- and as a consequence, it receives a reward R_{t+1},
- and finds itself in a new state S_{t+1}.

Reinforcement Learning Schema

Agent/system and environment/user interact

- at each of a sequence of time steps $t = 0, 1, 2, \ldots$,
- where agent receives a state representation S_t,
- on which basis it selects an action A_t,
- and as a consequence, it receives a reward R_{t+1},
- and finds itself in a new state S_{t+1}.

Goal of RL: Maximize the total amount of reward an agent receives in such interactions in the long run.

Formalizing User/Environment: Markov Decision Processes (MDPs)

A **Markov decision process** is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ where

- $\mathcal{S}$ is a set of states,
- $\mathcal{A}$ is a finite set of actions,
- $\mathcal{P}$ is a state transition probability matrix s.t. $\mathcal{P}_{ss'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a]$,
- $\mathcal{R}$ is a reward function s.t. $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$.

Dynamics of MDPs

One-step dynamics of the environment under the Markov property is completely specified by probability distribution over pairs of next state and reward s', r, given state and action s, a:

- $p(s', r|s, a) = P[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a]$.

Dynamics of MDPs

One-step dynamics of the environment under the Markov property is completely specified by probability distribution over pairs of next state and reward s', r, given state and action s, a:

- $p(s', r|s, a) = P[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a]$.

Exercise: Specify $\mathcal{P}^a_{ss'}$ and $\mathcal{R}^a_s$ in terms of $p(s', r|s, a)$.

Dynamics of MDPs

One-step dynamics of the environment under the Markov property is completely specified by probability distribution over pairs of next state and reward s', r, given state and action s, a:

- $p(s', r|s, a) = P[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a]$.

Exercise: Specify $\mathcal{P}^a_{ss'}$ and $\mathcal{R}^a_s$ in terms of $p(s', r|s, a)$.
$\mathcal{P}^a_{ss'} = p(s'|s, a) = \sum_{r \in \mathcal{R}} p(s', r|s, a)$,

Dynamics of MDPs

One-step dynamics of the environment under the Markov property is completely specified by probability distribution over pairs of next state and reward s', r, given state and action s, a:

- $p(s', r|s, a) = P[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a]$.

Exercise: Specify $\mathcal{P}_{ss'}^a$ and $\mathcal{R}_s^a$ in terms of $p(s', r|s, a)$.
$\mathcal{P}_{ss'}^a = p(s'|s, a) = \sum_{r\in\mathcal{R}} p(s', r|s, a)$,
$\mathcal{R}_s^a = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} p(s', r|s, a)$.

Formalizing Agent/System: Policies

A **stochastic policy** is a distribution over actions given states s.t.

- $\pi(a|s) = P[A_t = a|S_t = s]$.
- A policy completely specifies the behavior of an agent/system.
- Policies are parameterized π_θ, e.g. by a linear model or a neural nework - we use π to denote π_θ if unambiguous.
- Deterministic policies $a = \pi(s)$ also possible.

Policy Evaluation and Policy Optimization

Two central tasks in RL:

- **Policy evaluation (a.k.a. prediction):** Evaluate the expected reward for a given policy.
- **Policy optimization (a.k.a. learning/control):** Find the optimal policy / optimize a parametric policy under the expected reward criterion.

Return and Value Functions

- The **total discounted return** from time-step t for discount $\gamma \in [0, 1]$ is
  - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.

Return and Value Functions

- The **total discounted return** from time-step t for discount $\gamma \in [0, 1]$ is
  - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.
- The **action-value function** $q_\pi(s, a)$ on an MDP is the expected return starting from state s, taking action a, and following policy π s.t.
  - $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$.

Return and Value Functions

- The **total discounted return** from time-step t for discount $\gamma \in [0, 1]$ is
  - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.
- The **action-value function** $q_\pi(s, a)$ on an MDP is the expected return starting from state s, taking action a, and following policy π s.t.
  - $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$.
- The **state-value function** $v_\pi(s)$ on an MDP is the expected return starting from state s and following policy π s.t.
  - $v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_{a \sim \pi}[q_\pi(s, a)]$.

Bellman Expectation Equation

The state-value function can be decomposed into immediate reward plus discounted value of successor state s.t.

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right).
\end{aligned}
$$

Bellman Expectation Equation

The state-value function can be decomposed into immediate reward plus discounted value of successor state s.t.

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right).
\end{aligned}
$$

In matrix notation:

$$
\begin{aligned}
\mathbf{v}_\pi &= \boldsymbol{\mathcal{R}}^\pi + \gamma \boldsymbol{\mathcal{P}}^\pi \mathbf{v}_\pi \\
\text{where } \boldsymbol{\mathcal{R}}^\pi &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a, \\
\boldsymbol{\mathcal{P}}^\pi &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{ss'}^a.
\end{aligned}
$$

Policy Evaluation by Linear Programming

The value of $\mathbf{v}_\pi$ can be found directly by solving the linear equations of the Bellman Expectation Equation:

- **Solving linear equations:**

$$\mathbf{v}_\pi = (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

Policy Evaluation by Linear Programming

The value of $\mathbf{v}_\pi$ can be found directly by solving the linear equations of the Bellman Expectation Equation:

- **Solving linear equations:**

$$\mathbf{v}_\pi = (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

  - Only applicable to small MDPs.

Policy Evaluation by Linear Programming

The value of $\mathbf{v}_\pi$ can be found directly by solving the linear equations of the Bellman Expectation Equation:

- **Solving linear equations:**

$$\mathbf{v}_\pi = (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

  - Only applicable to small MDPs.

Exercise: Derive solution $\mathbf{v}_\pi$ from Bellman Expectation Equation.

Policy Evaluation by Linear Programming

The value of $\mathbf{v}_\pi$ can be found directly by solving the linear equations of the Bellman Expectation Equation:

- **Solving linear equations:**

$$\mathbf{v}_\pi = (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

  - Only applicable to small MDPs.

Exercise: Derive solution $\mathbf{v}_\pi$ from Bellman Expectation Equation.

$$\begin{aligned} \mathbf{v}_\pi &= \mathcal{R}^\pi + \gamma \mathcal{P}^\pi \mathbf{v}_\pi \\ (\mathbf{I} - \gamma \mathcal{P}^\pi) \mathbf{v}_\pi &= \mathcal{R}^\pi \\ \mathbf{v}_\pi &= (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi \end{aligned}$$

Policy Evaluation by Dynamic Programming (DP)

Value of $\mathbf{v}_\pi$ can also be found by iterative application of Bellman Expectation Equation:

- **Iterative policy evaluation:**

$$\mathbf{v}_{k+1} = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi \mathbf{v}_k .$$

Policy Evaluation by Dynamic Programming (DP)

Value of $\mathbf{v}_\pi$ can also be found by iterative application of Bellman Expectation Equation:

- **Iterative policy evaluation:**

$$\mathbf{v}_{k+1} = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi \mathbf{v}_k.$$

  - Performs **dynamic programming** by recursive decomposition of Bellman equation.
  - Can be parallelized (or backed up asynchronously), thus applicable to large MDPs.
  - Converges to $\mathbf{v}_\pi$.

Policy Optimization using Bellman Optimality Equation

An optimal policy π_* can be found by maximizing over the optimal action-value function $q_*(s,a) = \max_\pi q_\pi(s,a)$ s.t.

$$\pi_*(s) = \operatorname*{argmax}_a q_*(s,a).$$

Policy Optimization using Bellman Optimality Equation

An optimal policy π_* can be found by maximizing over the optimal action-value function $q_*(s, a) = \max_\pi q_\pi(s, a)$ s.t.

$$\pi_*(s) = \underset{a}{\operatorname{argmax}}\, q_*(s, a).$$

The optimal value functions are recursively related by the Bellman Optimality Equation:

$$\begin{aligned} q_*(s, a) &= \mathbb{E}_{\pi_*}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] \\ &= \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a'). \end{aligned}$$

Policy Optimization by Value Iteration

The Bellman Optimality Equation is non-linear and requires iterative solutions such as value iteration by dynamic programming:

- **Value iteration for q-function:**

$$q_{k+1}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_k(s',a').$$

  - Converges to $q_*(s,a)$.

Summary: Dynamic Programming

- Earliest RL algorithms with well-defined convergence properties.
- Bellman equation gives recursive decomposition for iterative solution to various problems in policy evaluation and policy optimization.
- Can be trivially parallelized or even run asynchronously.

Summary: Dynamic Programming

- Earliest RL algorithms with well-defined convergence properties.
- Bellman equation gives recursive decomposition for iterative solution to various problems in policy evaluation and policy optimization.
- Can be trivially parallelized or even run asynchronously.
- We **need to know a full MDP model** with all transitions and rewards, and touch all of them in learning!

Policy Evaluation by Monte-Carlo (MC) Sampling

- **Monte-Carlo Policy Evaluation**
  - Sample episodes $S_0, A_0, R_1, \ldots, R_T \sim \pi$.
  - For each sampled episode:
    - Increment state counter $N(s) \leftarrow N(s) + 1$.
    - Increment total return $S(s) \leftarrow S(s) + G_t$.
  - Estimate mean return $V(s) = S(s)/N(s)$.

Policy Evaluation by Monte-Carlo (MC) Sampling

- **Monte-Carlo Policy Evaluation**
  - Sample episodes $S_0, A_0, R_1, \ldots, R_T \sim \pi$.
  - For each sampled episode:
    - Increment state counter $N(s) \leftarrow N(s) + 1$.
    - Increment total return $S(s) \leftarrow S(s) + G_t$.
  - Estimate mean return $V(s) = S(s)/N(s)$.
- Learns v_π from episodes sampled under policy π, thus **model-free**.
- Updates can be done at first step or at every time step t where state s is visited in episode.
- Converges to v_π for large number of samples.

Incremental Mean

Use definition of incremental mean μ_k s.t.

$$\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right).
\end{aligned}$$

Incremental Monte-Carlo Updates

- **Incremental Monte-Carlo Policy Evaluation**
  - For each sampled episode, for each step t:
    - $N(S_t) \leftarrow N(S_t) + 1$,
    - $V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$.

Incremental Monte-Carlo Updates

- **Incremental Monte-Carlo Policy Evaluation**
  - For each sampled episode, for each step t:
    - $N(S_t) \leftarrow N(S_t) + 1$,
    - $V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$.
- Can be seen as **incremental update towards actual return**.
- α can be $\frac{1}{N(S_t)}$ or more general term $\alpha > 0$.

Policy Evaluation by Temporal Difference (TD) Learning

- **TD(0):**
  - For each sampled episode, for each step t:
    - $V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right).$

Policy Evaluation by Temporal Difference (TD) Learning

- **TD(0):**
  - For each sampled episode, for each step t:
    - $V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right).$
- **Combines sampling and recursive computation** by updating toward estimated return $R_{t+1} + \gamma V(S_{t+1})$.
- Recall $R_{t+1} + \gamma V(S_{t+1})$ from Bellman Expectation Equation, here called *TD target*.
- $\delta_t = (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$ is called *TD error*.

TD Learning with *n*-Step Returns

n-**Step Returns**:

- $G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$.
- $G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$.
- $\vdots$
- $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$.

TD Learning with *n*-Step Returns

n-**Step Returns**:

- $G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$.
- $G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$.
- $\vdots$
- $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$.

n-**Step TD Learning**:

- $V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$.

TD Learning with *n*-Step Returns

n-**Step Returns**:

- $G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$.
- $G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$.
- $\vdots$
- $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$.

n-**Step TD Learning**:

- $V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$.

Exercise: How can Incremental Monte Carlo be recovered by TD(n)?

TD Learning with *n*-Step Returns

n-**Step Returns**:

- $G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$.
- $G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$.
- $\vdots$
- $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$.

n-**Step TD Learning**:

- $V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$.

Exercise: How can Incremental Monte Carlo be recovered by TD(n)? Monte Carlo: $G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T$.

TD Learning with λ-Weighted Returns

λ-**Returns**:

- Average n-Step Returns using

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)},$$

where $\lambda \in [0,1]$.

TD Learning with λ-Weighted Returns

λ-**Returns**:

- Average n-Step Returns using

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)},$$

where $\lambda \in [0,1]$.

TD(λ) Learning:

- $V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{\lambda} - V(S_t) \right).$

TD Learning with λ-Weighted Returns

λ-**Returns**:

- Average n-Step Returns using

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)},$$

where $\lambda \in [0,1]$.

TD(λ) Learning:

- $V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^\lambda - V(S_t)\right).$

Exercise: How can TD(0) be recovered from TD(λ)?

TD Learning with λ-Weighted Returns

λ-**Returns**:

- Average n-Step Returns using

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)},$$

where $\lambda \in [0, 1]$.

TD(λ) Learning:

- $V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^\lambda - V(S_t) \right).$

Exercise: How can TD(0) be recovered from TD(λ)?
$\lambda = 0 \;\Rightarrow\; G_t^\lambda = G_t^{(1)} = TD(0).$

Policy Optimization by Q-Learning

- **Q-Learning** [Watkins and Dayan, 1992]:
- For each sampled episode:
  - Initialize S_t.
  - For each step t:
    - Sample A_t, observe R_{t+1}, S_{t+1}.
    - $Q(S_t, A_t) \leftarrow Q(S_t, A_t)$
      $+\alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$.
    - $S_t \leftarrow S_{t+1}$.

Policy Optimization by Q-Learning

- **Q-Learning** [Watkins and Dayan, 1992]:
- For each sampled episode:
  - Initialize S_t.
  - For each step t:
    - Sample A_t, observe R_{t+1}, S_{t+1}.
    - $Q(S_t, A_t) \leftarrow Q(S_t, A_t)$
      $+\alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$.
    - $S_t \leftarrow S_{t+1}$.
- **Q-Learning combines sampling and TD(0)-style recursive computation** for policy optimization.
- Recall $R_{t+1} + \gamma \max_{a'} Q(S_{t+1,a'})$ from Bellman Optimality Equation.

Summary: Monte-Carlo and Temporal-Difference Learning

- **MC** has **zero bias, but high variance** that grows with number of random actions, transitions, rewards in computation of return.

Summary: Monte-Carlo and Temporal-Difference Learning

- **MC** has **zero bias, but high variance** that grows with number of random actions, transitions, rewards in computation of return.
- **TD** techniques
  - **reduce variance** since TD target depends on a single random action, transition, reward,
  - can learn from **incomplete episodes** and can use **online updates**,
  - introduce **bias** and use approximations which are exact only in special cases.

Summary: Value-Based/Critic-Only Methods

- All techniques discussed so far, DP, MC, and TD, focus on **value-functions**, not policies.
- Value-function is also called **critic**, thus critic-only methods.
- Value-based techniques are inherently **indirect** in learning approximate value-function and extracting near-optimal policy.
- Overview over DP, MC, and TD in [Sutton and Barto, 1998] and [Szepesvári, 2009].

Summary: Value-Based/Critic-Only Methods

- All techniques discussed so far, DP, MC, and TD, focus on **value-functions**, not policies.
- Value-function is also called **critic**, thus critic-only methods.
- Value-based techniques are inherently **indirect** in learning approximate value-function and extracting near-optimal policy.
- Overview over DP, MC, and TD in [Sutton and Barto, 1998] and [Szepesvári, 2009].
- Problems:
  - Closeness to optimal policy cannot be quantified.
  - Focus is on deterministic instead of on stochastic policies.

Policy-Gradient Methods

- Policy-Gradient techniques attempt at **direct optimization of expected return**

  $$\mathbb{E}_{\pi_\theta}[G_t]$$

  for **parameterized stochastic policy**

  $$\pi_\theta(a|s) = P[A_t = a|S_t = s, \theta].$$

Policy-Gradient Methods

- Policy-Gradient techniques attempt at **direct optimization of expected return**

$$\mathbb{E}_{\pi_\theta}[G_t]$$

for **parameterized stochastic policy**

$$\pi_\theta(a|s) = P[A_t = a|S_t = s, \theta].$$

- Policy-function is also called **actor**.
- We will discuss **actor-only** (optimize parametric policy) and **actor-critic** (learn both policy and critic parameters in tandem) methods.

One-Step MDPs/Gradient Bandits

Let $p_\theta(y)$ denote probability of an action/output, $\Delta(y)$ be the reward/quality of an output.

$$\text{Objective: } \mathbb{E}_{p_\theta}[\Delta(y)]$$

$$\begin{aligned}
\text{Gradient: } \nabla_\theta \mathbb{E}_{p_\theta}[\Delta(y)] &= \nabla_\theta \sum_y p_\theta(y)\Delta(y) \\
&= \sum_y \nabla_\theta p_\theta(y)\Delta(y) \\
&= \sum_y \frac{p_\theta(y)}{p_\theta(y)} \nabla_\theta p_\theta(y)\Delta(y) \\
&= \sum_y p_\theta(y)\nabla_\theta \log p_\theta(y)\Delta(y) \\
&= \mathbb{E}_{p_\theta}[\Delta(y)\nabla_\theta \log p_\theta(y)].
\end{aligned}$$

Score Function Gradient Estimator for Bandit

- **Bandit Gradient Ascent:**
  - Sample $y_i \sim p_\theta$,
  - Update $\theta \leftarrow \theta + \alpha(\Delta(y_i)\nabla_\theta \log p_\theta(y_i))$.

Score Function Gradient Estimator for Bandit

- **Bandit Gradient Ascent:**
  - Sample $y_i \sim p_\theta$,
  - Update $\theta \leftarrow \theta + \alpha(\Delta(y_i)\nabla_\theta \log p_\theta(y_i))$.
- Update by stochastic gradient $g_i = \Delta(y_i)\nabla_\theta \log p_\theta(y_i)$ yields unbiased estimator of $\mathbb{E}_{p_\theta}[\Delta(y)]$

Score Function Gradient Estimator for Bandit

- **Bandit Gradient Ascent:**
  - Sample $y_i \sim p_\theta$,
  - Update $\theta \leftarrow \theta + \alpha(\Delta(y_i)\nabla_\theta \log p_\theta(y_i))$.
- Update by stochastic gradient $g_i = \Delta(y_i)\nabla_\theta \log p_\theta(y_i)$ yields unbiased estimator of $\mathbb{E}_{p_\theta}[\Delta(y)]$
- Intuition: $\nabla_\theta \log p_\theta(y)$ is called the **score function**.
  - Moving in the direction of g_i pushes up the score of the sample y_i in proportion to its reward $\Delta(y_i)$.
  - In RL terms: High reward samples are weighted higher - *reinforced*!
  - Estimator is valid even if $\Delta(y)$ is non-differentiable.

Score Function Gradient Estimator for MDPs

Let $y = S_0, A_0, R_1, \ldots, R_T \sim \pi_\theta$ be an episode, and $R(y) = R_1 + \gamma R_2 + \ldots + \gamma^{T-1} R_T = \sum_{t=1}^{T} \gamma^{t-1} R_t$ be its total discounted reward.

- Objective: $\mathbb{E}_{\pi_\theta}[R(y)]$.
- Gradient: $\mathbb{E}_{\pi_\theta}[R(y) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t)]$.

Score Function Gradient Estimator for MDPs

Let $y = S_0, A_0, R_1, \ldots, R_T \sim \pi_\theta$ be an episode, and $R(y) = R_1 + \gamma R_2 + \ldots + \gamma^{T-1} R_T = \sum_{t=1}^{T} \gamma^{t-1} R_t$ be its total discounted reward.

- Objective: $\mathbb{E}_{\pi_\theta}[R(y)]$.
- Gradient: $\mathbb{E}_{\pi_\theta}[R(y) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t)]$.
- **Reinforcement Gradient Ascent:**
  - Sample episode $y = S_0, A_0, R_1, \ldots, R_T \sim \pi_\theta$,
  - Obtain reward $R(y) = \sum_{t=1}^{T} \gamma^{t-1} R_t$,
  - Update $\theta \leftarrow \theta + \alpha(R(y) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t))$.

General Form of Policy Gradient Algorithms

Formalized for expected per time-step reward with respect to action-value $q_{\pi_\theta}(S_t, A_t)$.

- Objective: $\mathbb{E}_{\pi_\theta}[q_{\pi_\theta}(S_t, A_t)]$.
- Gradient: $\mathbb{E}_{\pi_\theta}[q_{\pi_\theta}(S_t, A_t)\nabla_\theta \log \pi_\theta(A_t|S_t)]$.

General Form of Policy Gradient Algorithms

Formalized for expected per time-step reward with respect to action-value $q_{\pi_\theta}(S_t, A_t)$.

- Objective: $\mathbb{E}_{\pi_\theta}[q_{\pi_\theta}(S_t, A_t)]$.
- Gradient: $\mathbb{E}_{\pi_\theta}[q_{\pi_\theta}(S_t, A_t)\nabla_\theta \log \pi_\theta(A_t|S_t)]$.
- **Policy Gradient Ascent**:
  - Sample episode $y = S_0, A_0, R_1, \ldots, R_T \sim \pi_\theta$.
  - For each time step t:
    - Obtain reward $q_{\pi_\theta}(S_t, A_t)$,
    - Update $\theta \leftarrow \theta + \alpha(q_{\pi_\theta}(S_t, A_t)\nabla_\theta \log \pi_\theta(A_t|S_t))$.

Policy Gradient Algorithms

- General form for expected per time-step return $q_{\pi_\theta}(S_t, A_t)$ is known as **Policy Gradient Theorem** [Sutton et al., 2000].
- Since $q_{\pi_\theta}(s, a)$ is normally not known, one can use the actual discounted return G_t at time step t, calculated from sampled episode. This leads to the **REINFORCE** algorithm [Williams, 1992].

Policy Gradient Algorithms

- General form for expected per time-step return $q_{\pi_\theta}(S_t, A_t)$ is known as **Policy Gradient Theorem** [Sutton et al., 2000].
- Since $q_{\pi_\theta}(s, a)$ is normally not known, one can use the actual discounted return G_t at time step t, calculated from sampled episode. This leads to the **REINFORCE** algorithm [Williams, 1992].
- Problems of Policy Gradient Algorithms, esp. REINFORCE:
    - Large variance in discounted returns calculated from sampled episodes.
    - Each gradient update is done independently of past gradient estimates.

Variance Reduction by Baselines

- Variance of REINFORCE can be reduced by comparison of actual return G_t to a baseline $b(s)$ for state s that is constant with respect to actions a. Example: average return so far.
- Update :

$$\theta \leftarrow \theta + \alpha(G_t - b(S_t))\nabla_\theta \log \pi_\theta(A_t|S_t)).$$

Variance Reduction by Baselines

- Variance of REINFORCE can be reduced by comparison of actual return G_t to a baseline $b(s)$ for state s that is constant with respect to actions a. Example: average return so far.
- Update :

$$\theta \leftarrow \theta + \alpha(G_t - b(S_t))\nabla_\theta \log \pi_\theta(A_t|S_t)).$$

- Can be interpreted as **Control Variate** [Ross, 2013]:
  - Goal is to augment random variable X (= stochastic gradient) with highly correlated variable Y such that $\mathrm{Var}(X - Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) - 2\mathrm{Cov}(X, Y)$ is reduced.
  - Gradient remains unbiased since $\mathbb{E}[X - Y + \mathbb{E}[Y]] = \mathbb{E}[X]$.

Variance Reduction by Baselines

Exercise: Show that $\mathbb{E}[Y] = 0$ for constant baselines.

Variance Reduction by Baselines

Exercise: Show that $\mathbb{E}[Y] = 0$ for constant baselines.
Proof:

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) b(s)] &= \sum_a \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} b(s) \\
&= b(s) \nabla_\theta \sum_a \pi_\theta(a|s) \\
&= b(s) \nabla_\theta 1 \\
&= 0.
\end{aligned}$$

Actor-Critic Methods

- Learning a critic in order to get an improved estimate of the expected return will also reduce variance.
  - **Critic:** $TD(0)$ update for linear approximation $q_{\pi_\theta}(s,a) \approx q_w(s,a) = \phi(s,a)^\top w$.
  - **Actor:** Policy gradient update reinforced by $q_w(s,a)$.

Actor-Critic Methods

- Learning a critic in order to get an improved estimate of the expected return will also reduce variance.
  - **Critic:** $TD(0)$ update for linear approximation $q_{\pi_\theta}(s,a) \approx q_w(s,a) = \phi(s,a)^\top w$.
  - **Actor:** Policy gradient update reinforced by $q_w(s,a)$.

- **Simple Actor-Critic** [Konda and Tsitsiklis, 2000]:
  - Sample $a \sim \pi_\theta$.
  - For each step t:
    - Sample reward $r \sim \mathcal{R}_s^a$, transition $s' \sim \mathcal{P}_{s,\cdot}^a$, action $a' \sim \pi_\theta(s', \cdot)$,
    - $\delta \leftarrow r + \gamma q_w(s', a') - q_w(s,a)$,
    - $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a|s) q_w(s,a)$,
    - $w \leftarrow w + \beta \delta \phi(s,a)$,
    - $a \leftarrow a', s \leftarrow s'$.

- True online updates of policy π_θ in each step!

Advantage Actor-Critic

- Combine idea of baseline with actor-critic by using **advantage function** that compares action-value function $q_{\pi_\theta}(s, a)$ to state-value function $v_{\pi_\theta}(s) = \mathbb{E}_{a \sim \pi}[q_{\pi_\theta}(s, a)]$.
- Use approximate TD error

$$\delta_w = r + \gamma v_w(s') - v_w(s),$$

where state-value is approximated by $v_w(s)$, and action-value is approximated by sample $q_w(s') = r + \gamma v_w(s')$.

Advantage Actor-Critic

- Combine idea of baseline with actor-critic by using **advantage function** that compares action-value function $q_{\pi_\theta}(s, a)$ to state-value function $v_{\pi_\theta}(s) = \mathbb{E}_{a\sim\pi}[q_{\pi_\theta}(s, a)]$.
- Use approximate TD error

$$\delta_w = r + \gamma v_w(s') - v_w(s),$$

where state-value is approximated by $v_w(s)$, and action-value is approximated by sample $q_w(s') = r + \gamma v_w(s')$.

- Update Actor: $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a|s)(q_w(s') - v_w(s))$.

Advantage Actor-Critic

- Combine idea of baseline with actor-critic by using **advantage function** that compares action-value function $q_{\pi_\theta}(s, a)$ to state-value function $v_{\pi_\theta}(s) = \mathbb{E}_{a\sim\pi}[q_{\pi_\theta}(s, a)]$.
- Use approximate TD error

$$\delta_w = r + \gamma v_w(s') - v_w(s),$$

  where state-value is approximated by $v_w(s)$, and action-value is approximated by sample $q_w(s') = r + \gamma v_w(s')$.
- Update Actor: $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a|s)(q_w(s') - v_w(s))$.
- Update Critic: $w = \arg\min_w (q_w(s') - v_w(s))^2$.

Summary: Policy-Gradient Methods

- Build upon huge knowlegde in stochastic optimization which provides **excellent theoretical understanding of convergence properties**.
- Gradient-based techniques are **model-free** since MDP transation matrix is not dependent on θ.
- Problem of **high variance** in **actor-only** methods can be mitigated by the **critic's low-variance estimate** of expected return.

Quick Summary and Outlook

What have we covered:

- **Policy evaluation (a.k.a. prediction)** using **DP**
- **Policy optimization (a.k.a. control)** using **Value-based** techniques of **DP**, **MC**, or both: **TD**.
- **Policy-gradient** techniques for direct stochastic optimization of parametric policies.

Quick Summary and Outlook

What have we covered:

- **Policy evaluation (a.k.a. prediction)** using **DP**
- **Policy optimization (a.k.a. control)** using **Value-based** techniques of **DP**, **MC**, or both: **TD**.
- **Policy-gradient** techniques for direct stochastic optimization of parametric policies.

Where from here on:

- **Sequence-to-Sequence** Reinforcement Learning
  - Algorithms for seq2seq RL from **simulated feedback**
  - Algorithms for offline learning from **logged feedback**
  - Seq2seq RL from **human bandit feedback**

Sequence-to-Sequence RL

Sequence-to-sequence (seq2seq) learning:

- $\mathbf{x} = x_1 \dots x_S$ represents an input sequence, indexed over a source vocabulary $\mathcal{V}_{\mathsf{Src}}$.
- $\mathbf{y} = y_1 \dots y_T$ represents an output sequence, indexed over a target vocabulary $\mathcal{V}_{\mathsf{Trg}}$.
- Goal of seq2seq learning is to estimate a function for mapping an input sequence $\mathbf{x}$ into an output sequences $\mathbf{y}$, defined as product of conditional token probabilities:

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid \mathbf{x}; \mathbf{y}_{<t}).$$

Seq2seq RL: Neural Machine Translation

Neural machine translation (NMT):

- $\mathbf{x}$ are source sentences, $\mathbf{y}$ are human reference translations,
- **Maximize likelihood of parallel data** $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$:

$$L(\theta) = \sum_{i=1}^{n} \log p_\theta(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)})$$

- $p_\theta(y_t \mid \mathbf{x}; \mathbf{y}_{<t})$ is defined by the neural model's softmax-normalized output vector of size $\mathbb{R}^{|\mathcal{V}_{\mathrm{Trg}}|}$:

$$p_\theta(y_t \mid \mathbf{x}; \mathbf{y}_{<t}) = \mathtt{softmax}(\mathrm{NN}_\theta(\mathbf{x}; \mathbf{y}_{<t})).$$

Seq2seq RL: Neural Machine Translation

Neural machine translation (NMT):

- $\mathbf{x}$ are source sentences, $\mathbf{y}$ are human reference translations,
- **Maximize likelihood of parallel data** $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$:

$$L(\theta) = \sum_{i=1}^{n} \log p_\theta(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)})$$

- $p_\theta(y_t \mid \mathbf{x}; \mathbf{y}_{<t})$ is defined by the neural model's softmax-normalized output vector of size $\mathbb{R}^{|\mathcal{V}_{\mathrm{Trg}}|}$:

$$p_\theta(y_t \mid \mathbf{x}; \mathbf{y}_{<t}) = \mathtt{softmax}(\mathrm{NN}_\theta(\mathbf{x}; \mathbf{y}_{<t})).$$

- Various options for NN_θ, such as recurrent [Sutskever et al., 2014, Bahdanau et al., 2015], convolutional [Gehring et al., 2017] or attentional [Vaswani et al., 2017] encoder-decoder architectures (or mix [Chen et al., 2018]).

Seq2seq RL for NMT

Why deviate from supervised learning using parallel data?

Seq2seq RL for NMT

Why deviate from supervised learning using parallel data?

- What if **no human references** are available, e.g., in under-resourced language pairs?
- Maybe **weak human feedback signals are easier to obtain** than full translations, e.g., from logged user interactions in commercial NMT services?
- [Sutton and Barto, 2018] on the "Future of Artificial Intelligence":

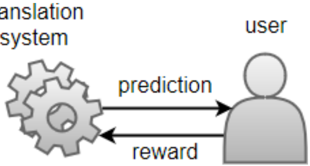
  The full potential of reinforcement learning requires reinforcement learning agents to be embedded into the flow of real-world experience, where they act, explore, and learn in our world, not just in their worlds.

Seq2seq RL for NMT

- Learning from weak user feedback in form of user clicks is state-of-the-art in computational advertising [Bottou et al., 2013, Chapelle et al., 2014].

Seq2seq RL for NMT

- Learning from weak user feedback in form of user clicks is state-of-the-art in computational advertising [Bottou et al., 2013, Chapelle et al., 2014].
- Let's dig the **gold mine of user feedback** to improve NMT!



Collecting Feedback: Facebook

Collecting Feedback: Facebook

Collecting Feedback: Facebook

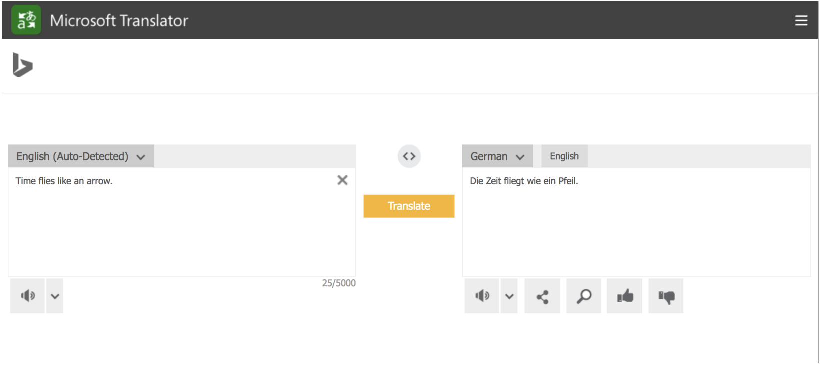
Collecting Feedback: Facebook

Collecting Feedback: Facebook

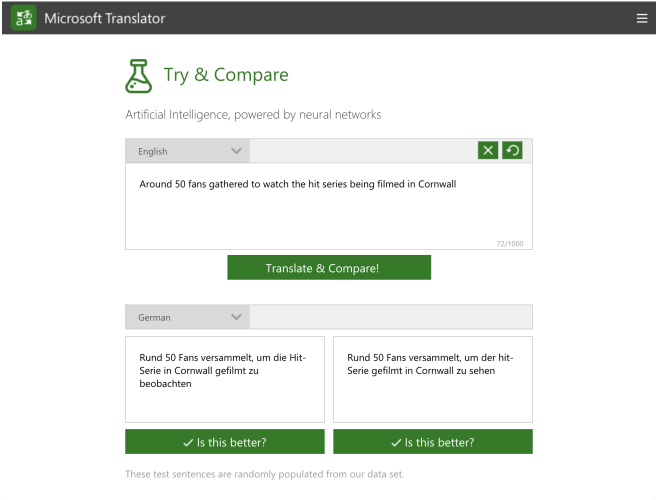
Collecting Feedback: Facebook

Collecting Feedback: Facebook

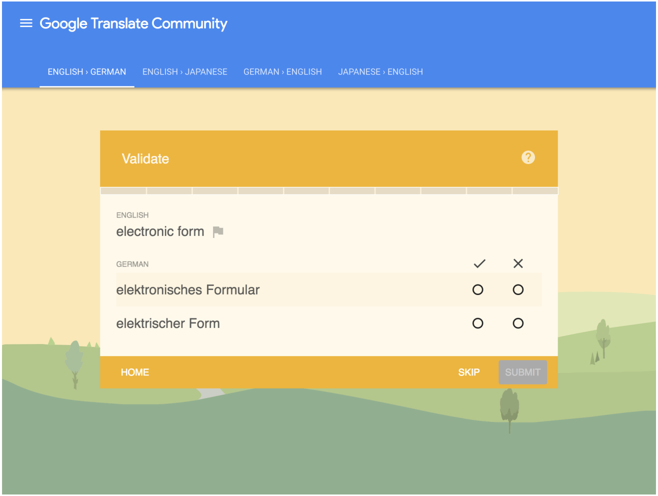
Collecting Feedback: Microsoft



Collecting Feedback: Microsoft (community)



Collecting Feedback: Google (community)



Collecting Feedback: Google

Seq2seq RL for NMT: Simulations

- NMT in standard RL framework:
  - In timestep t, a **state** is defined by the input $\mathbf{x}$ and the currently produced tokens $\tilde{\mathbf{y}}_{<t}$.
  - A **reward** is obtained by evaluating quality of the fully generated sequence $\tilde{\mathbf{y}}$.
  - An **action** corresponds to generating the next token $\tilde{y}_t$.

Seq2seq RL for NMT: Simulations

- NMT in standard RL framework:
  - In timestep t, a **state** is defined by the input $\mathbf{x}$ and the currently produced tokens $\tilde{\mathbf{y}}_{<t}$.
  - A **reward** is obtained by evaluating quality of the fully generated sequence $\tilde{\mathbf{y}}$.
  - An **action** corresponds to generating the next token $\tilde{y}_t$.
- Exercise: How would this translate into an MDP's state transitions and an agent's policy?

Seq2seq RL for NMT: Simulations

- NMT in standard RL framework:
  - In timestep t, a **state** is defined by the input $\mathbf{x}$ and the currently produced tokens $\tilde{\mathbf{y}}_{<t}$.
  - A **reward** is obtained by evaluating quality of the fully generated sequence $\tilde{\mathbf{y}}$.
  - An **action** corresponds to generating the next token $\tilde{y}_t$.
- Exercise: How would this translate into an MDP's state transitions and an agent's policy?
  - $p_\theta(\tilde{y}_t \mid \mathbf{x}; \tilde{\mathbf{y}}_{<t})$ corresponds to a **stochastic policy**, while the **state transition is deterministic** given an action.

Seq2seq RL for NMT: Simulations

- NMT in standard RL framework:
  - In timestep t, a **state** is defined by the input $\mathbf{x}$ and the currently produced tokens $\tilde{\mathbf{y}}_{<t}$.
  - A **reward** is obtained by evaluating quality of the fully generated sequence $\tilde{\mathbf{y}}$.
  - An **action** corresponds to generating the next token $\tilde{y}_t$.
- Exercise: How would this translate into an MDP's state transitions and an agent's policy?
  - $p_\theta(\tilde{y}_t \mid \mathbf{x}; \tilde{\mathbf{y}}_{<t})$ corresponds to a **stochastic policy**, while the **state transition is deterministic** given an action.
- Interactive NMT:
  - The **NMT system is the agent** that performs actions, while the **human user provides rewards**.

Seq2seq RL for NMT: Simulations

- Expected loss/reward objective:

$$L(\theta) = \mathbb{E}_{p(\mathbf{x})\, p_\theta(\tilde{\mathbf{y}}|\mathbf{x};\theta)} \left[\Delta(\tilde{\mathbf{y}})\right]$$

  where $\Delta(\tilde{\mathbf{y}})$ is task loss, e.g., $-\mathrm{BLEU}(\tilde{\mathbf{y}})$
- Sampling an input $\mathbf{x}$ and an output $\tilde{\mathbf{y}}$, and performing a stochastic gradient descent update corresponds to a **policy gradient** algorithm.

(Neural) Bandit Structured Prediction

Algorithm 1 (Neural) Bandit Structured Prediction

1: **for** $k = 0, \ldots, K$ **do**
2: Observe input $\mathbf{x}_k$
3: Sample output $\tilde{\mathbf{y}}_k \sim p_\theta(\mathbf{y}|\mathbf{x}_k)$
4: Obtain feedback $\Delta(\tilde{\mathbf{y}}_k)$
5: Update parameters $\theta_{k+1} = \theta_k - \gamma_k \, s_k$
6: where stochastic gradient $s_k = \Delta(\tilde{\mathbf{y}}) \frac{\partial \log p_\theta(\tilde{\mathbf{y}}|\mathbf{x}_k)}{\partial \theta_i}$.

- [Sokolov et al., 2015, Sokolov et al., 2016, Kreutzer et al., 2017]

(Neural) Bandit Structured Prediction

- Why (Neural) **Bandit** Structured Prediction?
  - An action is defined as generating a full output sequence, thus corresponding to a **one-state MDP**.
  - Term **bandit feedback** is inherited from the problem of maximizing the reward for a sequence of pulls of arms of so-called "one-armed bandit" slot machines [Bubeck and Cesa-Bianchi, 2012]:
    - In contrast to fully supervised learning, the learner receives feedback to a single prediction. It does not know what the correct output looks like, nor what would have happened if it had predicted differently.
  - Related to gradient bandit algorithms [Sutton and Barto, 2018] and contextual bandits [Li et al., 2010].

(Neural) Bandit Structured Prediction

- Important measure for variance reduction: **Control variates**
  - Random variable X is stochastic gradient s_k in case of algorithm 1.
  - Two choices in [Kreutzer et al., 2017]:
    1. **Baseline** [Williams, 1992]:

$$Y_k = \nabla \log p_\theta(\tilde{\mathbf{y}}|\mathbf{x}_k) \, \frac{1}{k} \sum_{j=1}^{k} \Delta(\tilde{\mathbf{y}}_j).$$

    2. **Score Function** [Ranganath et al., 2014]:

$$Y_k = \nabla \log p_\theta(\tilde{\mathbf{y}}|\mathbf{x}_k).$$

Advantage Actor-Critic for Bandit NMT

- Neural encoder-decoder A2C [Nguyen et al., 2017]:
  - Gradient approximation

$$\nabla L(\theta) \approx \sum_{t=1}^{T} \bar{R}_t(\tilde{\mathbf{y}}) \nabla_\theta \log p_\theta(\tilde{y}_t \mid \mathbf{x}; \tilde{\mathbf{y}}_{<t})$$

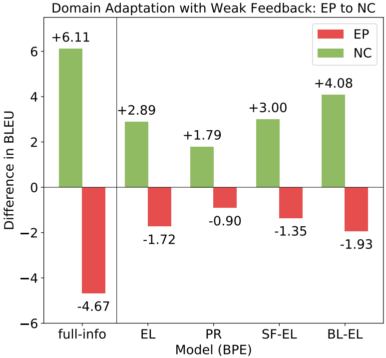
  - Uses **per-action advantage function**

$$\bar{R}_t(\tilde{\mathbf{y}}) := \Delta(\tilde{\mathbf{y}}) - V(\tilde{\mathbf{y}}_{<t})$$

  - State-value function $V(\tilde{\mathbf{y}}_{<t})$ centers the reward and uses separate neural encoder-decoder network that is trained to minimize the squared error $[V_w(\tilde{\mathbf{y}}_{<t}) - \Delta(\tilde{\mathbf{y}})]^2$

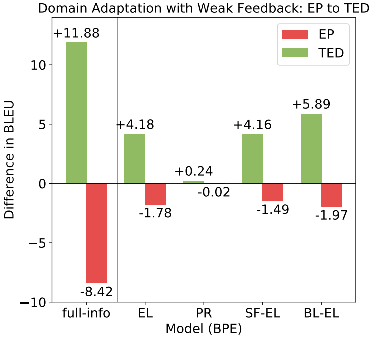
Seq2seq RL for NMT: Simulation Results

- EuroParl→NewsComm NMT conservative domain adaptation
- $\Delta(\tilde{\mathbf{y}})$ simulated by per-sentence BLEU against reference



Seq2seq RL for NMT: Simulation Results

- EuroParl→TED NMT conservative domain adaptation task



Seq2seq RL for NMT: To Simulate or Not

- **Domain adaptation** experiments show **impressive gains** for learning from simulated bandit feedback only
- Most work on Seq2seq RL for NMT is **confined to simulations**, aiming to improve "exposure bias" and "loss-evaluation mismatch" [Ranzato et al., 2016]
- Recall [Sutton and Barto, 2018] on the "Future of Artificial Intelligence":

> *A major reason for wanting a reinforcement learning agent to act and learn in the real world is that it is often difficult, sometimes impossible, to simulate real-world experience with enough fidelity to make the resulting policies [...] work well—and safely—when directing real actions.*

Seq2seq RL for NMT: From Simulations to Human RL

- Where do simulations fall short?
  - Real-world RL only has access to **human bandit feedback** to a single prediction—no summation over all actions that amounts to full supervision [Shen et al., 2016, Bahdanau et al., 2017].
  - Online/on-policy learning might be undesirable given concerns about **safety and stability of commercial systems**.
  - **Reward function** for human translation quality is **not well defined**, reward signals are **noisy and skewed**.

Seq2seq RL for NMT: From Simulations to Human RL

- Where do simulations fall short?
  - Real-world RL only has access to **human bandit feedback** to a single prediction—no summation over all actions that amounts to full supervision [Shen et al., 2016, Bahdanau et al., 2017].
  - Online/on-policy learning might be undesirable given concerns about **safety and stability of commercial systems**.
  - **Reward function** for human translation quality is **not well defined**, reward signals are **noisy and skewed**.
- (Super)human performance (similar to playing Atari or Go) of real-world RL is not to be expected soon!

Offline Learning from Logged Feedback

Standard: Online/On-Policy RL

- Undesirable if stability or real-world system has priority over frequent updates after each interaction

Offline Learning from Logged Feedback

Standard: Online/On-Policy RL

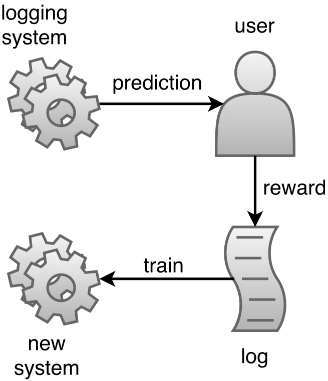
- Undesirable if stability or real-world system has priority over frequent updates after each interaction

Offline/Off-Policy RL from Logged Bandit Feedback

- Attempts to learn from logged feedback that has been given to the predictions of a historic system following a different policy
- Allows control over system updates
- Prior work in counterfactual bandit learning [Dudik et al., 2011, Bottou et al., 2013] and off-policy RL [Precup et al., 2000, Jiang and Li, 2016]

Offline Learning = Counterfactual Learning

- Counterfactual question: Estimate how the new system would have performed if it had been in control of choosing the logged predictions.



Offline Learning from Logged Feedback

- Logged data $D = \{(\mathbf{x}^{(h)}, \mathbf{y}^{(h)}, r(\mathbf{y}^{(h)}))\}_{h=1}^{H}$ where $\mathbf{y}^{(h)}$ is sampled from a logging system $\mu(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})$, and the reward/loss $r(\mathbf{y}^{(h)}) \in [0,1]$ is obtained from human user.
- Inverse propensity scoring (IPS) to learn target policy $p_\theta(\mathbf{y}|\mathbf{x})$:

$$L(\theta) = \frac{1}{H} \sum_{h=1}^{H} r(\mathbf{y}^{(h)})\, \rho_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}).$$

- IPS uses **importance sampling** to correct for sampling bias of logging system s.t. $\rho_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}) = \frac{p_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}{\mu(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}$

Offline Learning from Logged Feedback

- Logged data $D = \{(\mathbf{x}^{(h)}, \mathbf{y}^{(h)}, r(\mathbf{y}^{(h)}))\}_{h=1}^{H}$ where $\mathbf{y}^{(h)}$ is sampled from a logging system $\mu(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})$, and the reward/loss $r(\mathbf{y}^{(h)}) \in [0, 1]$ is obtained from human user.
- Inverse propensity scoring (IPS) to learn target policy $p_\theta(\mathbf{y}|\mathbf{x})$:

$$L(\theta) = \frac{1}{H} \sum_{h=1}^{H} r(\mathbf{y}^{(h)})\, \rho_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}).$$

- IPS uses **importance sampling** to correct for sampling bias of logging system s.t. $\rho_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}) = \frac{p_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}{\mu(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}$
- Exercise: Show unbiasedness of IPS estimator.

Offline Learning from Logged Feedback

- Logged data $D = \{(\mathbf{x}^{(h)}, \mathbf{y}^{(h)}, r(\mathbf{y}^{(h)}))\}_{h=1}^{H}$ where $\mathbf{y}^{(h)}$ is sampled from a logging system $\mu(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})$, and the reward/loss $r(\mathbf{y}^{(h)}) \in [0,1]$ is obtained from human user.
- Inverse propensity scoring (IPS) to learn target policy $p_\theta(\mathbf{y}|\mathbf{x})$:

$$L(\theta) = \frac{1}{H} \sum_{h=1}^{H} r(\mathbf{y}^{(h)})\, \rho_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}).$$

- IPS uses **importance sampling** to correct for sampling bias of logging system s.t. $\rho_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}) = \frac{p_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}{\mu(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}$
- Exercise: Show unbiasedness of IPS estimator.

$$\frac{1}{H} \sum_{h=1}^{H} r(\mathbf{y}^{(h)}) \frac{p_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}{\mu(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})} = \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{\mu(\mathbf{y}|\mathbf{x})}[r(\mathbf{y}) \frac{p_\theta(\mathbf{y}|\mathbf{x})}{\mu(\mathbf{y}|\mathbf{x})}]$$
$$= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})}[r(\mathbf{y})].$$

Offline Learning under Deterministic Logging: Problems

- Commercial NMT systems try to avoid risk by showing only most probable translation to users = exploration-free, deterministic logging

Offline Learning under Deterministic Logging: Problems

- Commercial NMT systems try to avoid risk by showing only most probable translation to users = exploration-free, deterministic logging
- Problems with deterministic logging [Lawrence et al., 2017a]
  - **No correction of sampling bias** like in IPS since $\mu(\mathbf{y}|\mathbf{x}) = 1$
  - **Degenerate behavior**: Empirical reward over log is maximized by setting probability of *all* logged data to 1 $\rightarrow$ Undesirable to increase probability of low reward examples
  - Unbiased learning is **thought to be impossible** for exploration-free off-policy learning [Langford et al., 2008, Strehl et al., 2010].

Offline Learning under Deterministic Logging: Solutions

- **Implicit exploration** via inputs [Bastani et al., 2017]

Offline Learning under Deterministic Logging: Solutions

- **Implicit exploration** via inputs [Bastani et al., 2017]
- **Deterministic Propensity Matching (DPM)** [Lawrence et al., 2017b, Lawrence and Riezler, 2018]

$$L(\theta) = \frac{1}{H} \sum_{h=1}^{H} r(\mathbf{y}^{(h)})\, \bar{p}_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}),$$

  - **Reweighting** by multiplicative control variate , evaluated **one-step-late** at θ' from some previous iteration: $\bar{p}_{\theta,\theta'}(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}) = \frac{p_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}{\sum_{b=1}^{B} p_{\theta'}(\mathbf{y}^{(b)}|\mathbf{x}^{(b)})}$.
  - **Effect of self-normalization:** Introduces bias that decreases as B increases [Kong, 1992], but prevents increasing probability for low reward data by taking away probability mass from higher reward outputs.

Offline Learning under Deterministic Logging: Gradients

- Optimization by Stochastic Gradient Descent
  - IPS:

$$\nabla L(\theta) = \frac{1}{H} \sum_{h=1}^{H} r(\mathbf{y}^{(h)})\, \rho_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}) \nabla \log p_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})$$

  - OSL self-normalized deterministic propensity matching:

$$\nabla L(\theta) = \frac{1}{H} \sum_{h=1}^{H} r(\mathbf{y}^{(h)})\, \bar{p}_{\theta,\theta'}(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}) \nabla \log p_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})$$

Seq2seq RL for NMT: From Simulations to Human RL

- Where do simulations fall short?
  - Real-wold RL only has access to **human bandit feedback**
  - Online/on-policy learning raises **safety and stability concerns**
  - **Human rewards** are **not well defined, noisy, and skewed**

Seq2seq RL for NMT: From Simulations to Human RL

- Where do simulations fall short?
  - Real-wold RL only has access to **human bandit feedback**
    $\Rightarrow$ **control variates**
  - Online/on-policy learning raises **safety and stability concerns**
  - **Human rewards** are **not well defined, noisy, and skewed**

Seq2seq RL for NMT: From Simulations to Human RL

- Where do simulations fall short?
  - Real-wold RL only has access to **human bandit feedback**
    $\Rightarrow$ **control variates**
  - Online/on-policy learning raises **safety and stability concerns**
    $\Rightarrow$ **offline learning**
  - **Human rewards** are **not well defined, noisy, and skewed**

Seq2seq RL for NMT: From Simulations to Human RL

- Where do simulations fall short?
  - Real-wold RL only has access to **human bandit feedback**
    $\Rightarrow$ **control variates**
  - Online/on-policy learning raises **safety and stability concerns**
    $\Rightarrow$ **offline learning**
  - **Human rewards** are **not well defined, noisy, and skewed**
    $\Rightarrow$ **reward estimation**

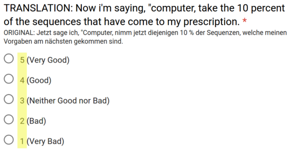
Offline Learning from Human Feedback: e-commerce

- [Kreutzer et al., 2018]: 69k translated item titles (en-es) with 148k individual ratings
- No agreement of paid raters with e-commerce users, low inter-rater agreement, learning impossible

Offline Learning from Human Feedback: e-commerce

- Lessons from e-commerce experiments:
    - Offline learning from direct user feedback to e-commerce titles is equivalent to **learning from noise**
    - Conjecture: Missing reliability and validity of human feedback in e-commerce experiment
    - Need experiment on controlled feedback collection!

Offline Learning from Controlled Human Feedback

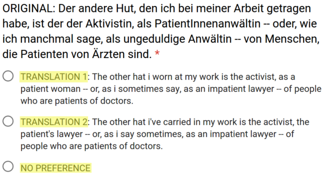


TRANSLATION: Now i'm saying, "computer, take the 10 percent of the sequences that have come to my prescription. *

ORIGINAL: Jetzt sage ich, "Computer, nimm jetzt diejenigen 10 % der Sequenzen, welche meinen Vorgaben am nächsten gekommen sind.

- 5 (Very Good)
- 4 (Good)
- 3 (Neither Good nor Bad)
- 2 (Bad)
- 1 (Very Bad)

vs



ORIGINAL: Der andere Hut, den ich bei meiner Arbeit getragen habe, ist der der Aktivistin, als PatientInnenanwältin -- oder, wie ich manchmal sage, als ungeduldige Anwältin -- von Menschen, die Patienten von Ärzten sind. *

- TRANSLATION 1: The other hat i worn at my work is the activist, as a patient woman -- or, as i sometimes say, as an impatient lawyer -- of people who are patients of doctors.
- TRANSLATION 2: The other hat i've carried in my work is the activist, the patient's lawyer -- or, as i say sometimes, as an impatient lawyer -- of people who are patients of doctors.
- NO PREFERENCE

- Ratings on five-point Likert scale (left) and pairwise preferences (right), ∼15 bilinguals for each task
- 800 de-en translations and 400 pairs[1], filtered for length 20-40 and paired by difference in chrF score [Popović, 2015]

[1] Data: https://www.cl.uni-heidelberg.de/statnlpgroup/humanmt/

Reliability and Learnability of Human Feedback

- Controlled study on main factors in human RL:
  1. **Reliability**: Collect five-point and pairwise feedback on same data, evaluate intra- and inter-rater agreement.
  2. **Learnability**: Train reward estimators on human feedback, evaluate correlation to TER on held-out data.
  3. **RL**: Use rewards directly or estimated rewards to improve an NMT system.

Reliability and Learnability of Human Feedback

- Controlled study on main factors in human RL:
  1. **Reliability**: Collect five-point and pairwise feedback on same data, evaluate intra- and inter-rater agreement.
  2. **Learnability**: Train reward estimators on human feedback, evaluate correlation to TER on held-out data.
  3. **RL**: Use rewards directly or estimated rewards to improve an NMT system.

What are your guesses on reliability and learnability—five-point or pairwise?

Reliability: α-agreement

Rating Type	Inter-rater α	Intra-rater Mean α	Intra-rater Stdev α
5-point	0.2308	0.4014	0.1907
+ normalization	0.2820		
+ filtering	**0.5059**	0.5527	0.0470
Pairwise	0.2385	0.5085	0.2096
+ filtering	0.3912	**0.7264**	0.0533

- Inter- and intra-reliability measured by Krippendorff's α for 5-point and pairwise ratings of 1,000 translations of which 200 translations are repeated twice.
- Filtered variants are restricted to either a subset of participants (5-point) or a subset of translations (pairwise).

Reliability: Qualitative Analysis

Rating Type	Avg. subjective difficulty [1-10]
5-point	4.8
Pairwise	5.69

- Difficulties with **5-point** ratings:
  - Weighing of error types; long sentences with few essential errors
- Difficulties with **Pairwise** ratings:
  - Distinction between similar translations
  - Ties: no absolute anchoring of the quality of the pair
  - Final score: No normalization for individual biases possible

Learnability: 5-point Feedback

- Inputs are sources $\mathbf{x}$ and their translations $\mathbf{y}$
- Given cardinal ratings r, train a regression model with parameters ψ to minimize the mean squared error (MSE) for predicted rewards $\hat{r}$:

$$L(\psi) = \frac{1}{n} \sum_{i=1}^{n} (r(\mathbf{y}_i) - \hat{r}_\psi(\mathbf{y}_i))^2.$$

Learnability: Pairwise Feedback

- Given human preference $Q[\mathbf{y^1} \succ \mathbf{y^2}]$ for translation $\mathbf{y_1}$ over translation $\mathbf{y_2}$
- Train estimator $\hat{P}_\psi[\mathbf{y^1} \succ \mathbf{y^2}]$ by minimizing cross-entropy between predictions and human preferences:

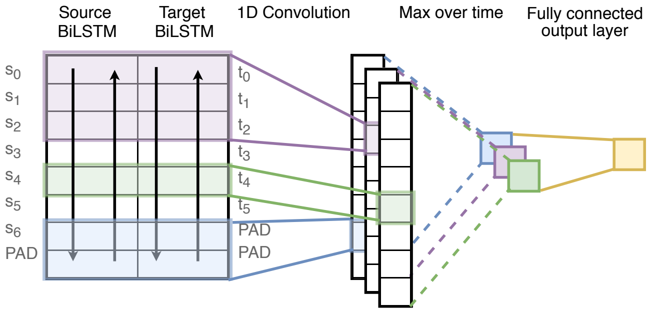
$$L(\psi) = -\frac{1}{n}\sum_{i=1}^{n}\Big(Q[\mathbf{y_i^1} \succ \mathbf{y_i^2}] \log \hat{P}_\psi[\mathbf{y_i^1} \succ \mathbf{y_i^2}] + Q[\mathbf{y_i^2} \succ \mathbf{y_i^1}] \log \hat{P}_\psi[\mathbf{y_i^2} \succ \mathbf{y_i^1}]\Big),$$

with the Bradley-Terry model for preferences

$$\hat{P}_\psi[\mathbf{y}^1 \succ \mathbf{y}^2] = \frac{\exp \hat{r}_\psi(\mathbf{y^1})}{\exp \hat{r}_\psi(\mathbf{y^1}) + \exp \hat{r}_\psi(\mathbf{y^2})}.$$

- Use Bradley-Terry model's $\hat{r}$ as reward estimator [Christiano et al., 2017]

Reward Estimator Architecture



- biLSTM-enhanced bilingual extension of convolutional model for sentence classification [Kim, 2014]

Learnability: Results

Model	**Feedback**	**Spearman's ρ with -TER**
MSE	5-point norm.	0.2193
	+ filtering	**0.2341**
PW	Pairwise	0.1310
	+ filtering	0.1255

- Comparatively better results for reward estimation from cardinal human judgements.
- Overall relatively low correlation, presumably due to overfitting on small training data set.

End-to-end Seq2seq RL

1. Tackle **the arguably simpler** problem of learning a reward estimator from human feedback first.
2. Then **provide unlimited learned feedback** to generalize to unseen outputs in off-policy RL.

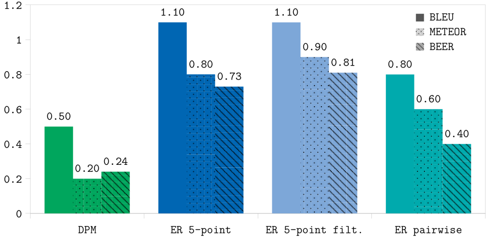
End-to-End RL from Estimated Rewards

Expected Risk Minimiziation from Estimated Rewards
Estimated rewards allow to use minimum risk training [Shen et al., 2016] s.t. feedback can be collected for k samples:

$$
\begin{aligned}
L(\theta) =& \mathbb{E}_{p(\mathbf{x})p_\theta(\mathbf{y}|\mathbf{x})}\left[\hat{r}_\psi(\mathbf{y})\right] \\
&\approx \sum_{s=1}^{S}\sum_{i=1}^{k} p_\theta^\tau(\tilde{\mathbf{y}}_{\mathbf{i}}^{(\mathbf{s})}|\mathbf{x}^{(\mathbf{s})})\,\hat{r}_\psi(\tilde{\mathbf{y}}_{\mathbf{i}})
\end{aligned}
$$

- Softmax temperature τ to control the amount of exploration by sharpening the sampling distribution $p_\theta^\tau(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(\mathbf{o}/\tau)$ at lower temperatures.
- Subtract the running average of rewards from $\hat{r}_\psi$ to reduce gradient variance and estimation bias.

Results on TED Talk Translations



- Significant improvements over the baseline (27.0 BLEU / 30.7 METEOR / 59.48 BEER):
  - Gains of 1.1 BLEU for expected risk (ER) minimization for estimated rewards.
  - Deterministic propensity matching (DPM) on directly logged human feedback yields up to 0.5 BLEU points.

Summary

Basic RL:

- **Policy evaluation** using **Dynamic Programming**
- **Policy optimization** using **Dynamic Programming**, **Monte Carlo**, or both: **Temporal Difference** learning.
- **Policy-gradient** techniques for direct policy optimization.

Seq2seq RL:

- Seq2seq RL **simulations**: Bandit Neural Machine Translation.
- **Offline** learning from deterministically logged feedback: Deterministic Propensity Matching.
- Seq2seq RL from **human feedback**: Collecting reliable feedback, learning reward estimators, end-to-end RL from estimated rewards.

Thank you!

Questions?

P.S.: I'm currently hiring PhD/PostDoc for projects on seq2seq RL for machine translation and conversational question-answering.

Email: riezler@cl.uni-heidelberg.de

Research: `https://www.cl.uni-heidelberg.de/statnlpgroup/`

References

- Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2017). An actor-critic algorithm for sequence prediction. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France.

- Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA.

- Bastani, H., Bayati, M., and Khosravi, K. (2017). Exploiting the natural exploration in contextual bandits. *ArXiv e-prints*, 1704.09011.

- Bottou, L., Peters, J., Quiñonero-Candela, J., Charles, D. X., Chickering, D. M., Portugaly, E., Ray, D., Simard, P., and Snelson, E. (2013). Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14:3207–3260.

- Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.

- Chapelle, O., Masnavoglu, E., and Rosales, R. (2014). Simple and scalable response prediction for display advertising.
  ACM Transactions on Intelligent Systems and Technology, 5(4).

- Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Schuster, M., Shazeer, N., Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Chen, Z., Wu, Y., and Hughes, M. (2018). The best of both worlds: Combining recent advances in neural machine translation.
  In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

- Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences.
  In *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA.

- Dudik, M., Langford, J., and Li, L. (2011). Doubly robust policy evaluation and learning.
  In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, WA.

- Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. (2017). Convolutional sequence to sequence learning.
  In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

- Jiang, N. and Li, L. (2016). Doubly robust off-policy value evaluation for reinforcement learning.
  In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, New York, NY.
- Kim, Y. (2014). Convolutional neural networks for sentence classification.
  In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.
- Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms.
  In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada.
- Kong, A. (1992). A note on importance sampling using standardized weights.
  Technical Report 348, Department of Statistics, University of Chicago, Illinois.
- Kreutzer, J., Khadivi, S., Matusov, E., and Riezler, S. (2018). Can neural machine translation be improved with user feedback?
  In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Industry Track (NAACL-HLT)*, New Orleans, LA.
- Kreutzer, J., Sokolov, A., and Riezler, S. (2017). Bandit structured prediction for neural sequence-to-sequence learning.
  In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

- Langford, J., Strehl, A., and Wortman, J. (2008). Exploration scavenging. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, Helsinki, Finland.
- Lawrence, C., Gajane, P., and Riezler, S. (2017a). Counterfactual learning for machine translation: Degeneracies and solutions. In *Proceedings of the NIPS WhatIF Workshop*, Long Beach, CA.
- Lawrence, C. and Riezler, S. (2018). Improving a neural semantic parser by counterfactual learning from human bandit feedback. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.
- Lawrence, C., Sokolov, A., and Riezler, S. (2017b). Counterfactual learning from bandit feedback under deterministic logging: A case study in statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark.
- Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*.
- Nguyen, K., Daumé, H., and Boyd-Graber, J. (2017). Reinforcement learning for bandit neural machine translation with simulated feedback.

In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark.

- Popović, M. (2015). chrF: character n-gram f-score for automatic mt evaluation.
  In *Proceedings of the Tenth Workshop on Statistical Machine Translat ion (WMT)*, Lisbon, Portugal.

- Precup, D., Sutton, R. S., and Singh, S. P. (2000). Eligibility traces for off-policy policy evaluation.
  In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, San Francisco, CA.

- Ranganath, R., Gerrish, S., and Blei, D. M. (2014). Black box variational inference.
  In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Reykjavik, Iceland.

- Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks.
  In *Proceedings of the International Conference on Learning Representation (ICLR)*, San Juan, Puerto Rico.

- Ross, S. M. (2013). *Simulation*.
  Elsevier, fifth edition.

- Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016). Minimum risk training for neural machine translation.
  In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany.
- Sokolov, A., Kreutzer, J., Lo, C., and Riezler, S. (2016). Stochastic structured prediction under bandit feedback.
  In *Advances in Neural Information Processing Systems (NIPS)*, Barcelona, Spain.
- Sokolov, A., Riezler, S., and Urvoy, T. (2015). Bandit structured prediction for learning from user feedback in statistical machine translation.
  In *Proceedings of MT Summit XV*, Miami, FL.
- Strehl, A. L., Langford, J., Li, L., and Kakade, S. M. (2010). Learning from logged implicit exploration data.
  In *Advances in Neural Information Processing Sytems (NIPS)*, Vancouver, Canada.
- Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks.
  In *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning. An Introduction*.
  The MIT Press.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning. An Introduction*.
  The MIT Press, second edition.

- Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation.
  In *Advances in Neural Information Processings Systems (NIPS)*, Vancouver, Canada.
- Szepesvári, C. (2009). *Algorithms for Reinforcement Learning*.
  Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.
  In *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA.
- Watkins, C. and Dayan, P. (1992). Q-learning.
  Machine Learning, 8:279–292.
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning.
  Machine Learning, 8:229–256.