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:

- $\mathbf{x} = x_1 \ldots x_S$ represents an input sequence, indexed over a source vocabulary $\mathcal{V}_{\mathsf{Src}}$.
- $\mathbf{y} = y_1 \ldots y_T$ 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 $\mathbb{R}^{|\mathcal{V}_{\mathrm{Trg}}|}$:

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

- 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

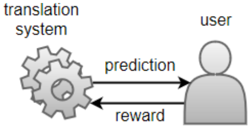
Why deviate from supervised learning using parallel data?

- 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].
- Let's dig the **gold mine of user feedback** to improve NMT!



Collecting Feedback: Facebook

Collecting Feedback: Facebook

Collecting Feedback: Facebook

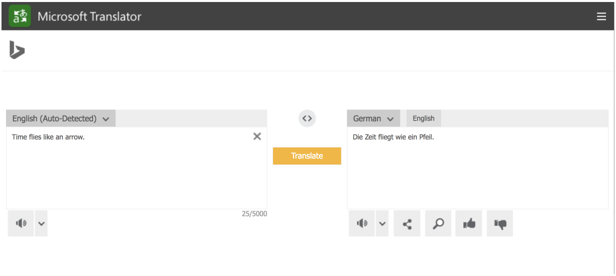
Collecting Feedback: Facebook

Collecting Feedback: Facebook

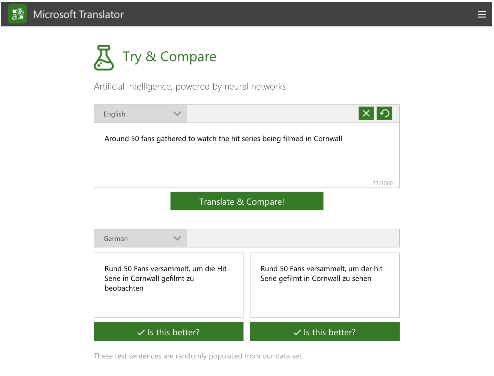
Collecting Feedback: Facebook

Collecting Feedback: Facebook

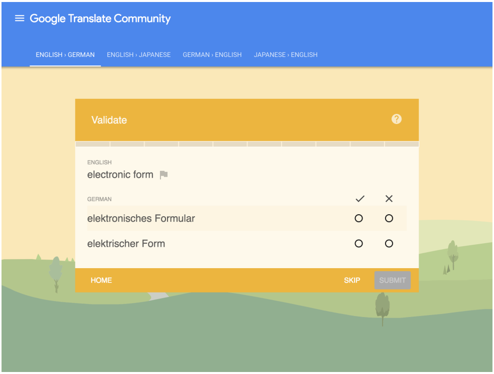
Collecting Feedback: Microsoft



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 $\tilde{y}_t$.
- 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

for $k = 0, \ldots, K$ **do**
    Observe input $\mathbf{x}_k$
    Sample output $\tilde{\mathbf{y}}_k \sim p_\theta(\mathbf{y}|\mathbf{x}_k)$
    Obtain feedback $\Delta(\tilde{\mathbf{y}}_k)$
    Update parameters $\theta_{k+1} = \theta_k - \gamma_k \, s_k$
        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

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

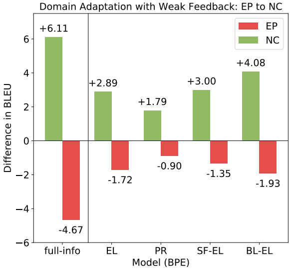
  - Uses **per-action advantage function**

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

  - 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 $[V_w(\tilde{\mathbf{y}}_{<t}) - \Delta(\tilde{\mathbf{y}})]^2$

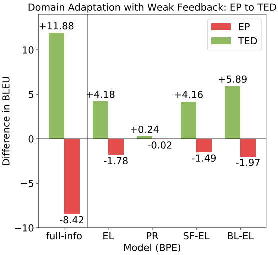
Seq2seq RL for NMT: Simulation Results

- 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: To Simulate or Not

- Where do simulations fall short?
  - Real-wold 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

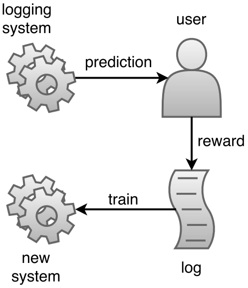
- Undesirable if stability or real-world system has priority over frequent updates after each interaction

Offline/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. $\rho_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}) = \frac{p_\theta(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}{\mu(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}$
- 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
- 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]
- **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 cariate , 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:

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