A Tutorial on Optimization for Neural Sequence Generation

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

Types of Prediction

- Two classes (**binary classification**)

  I hate this movie → positive / negative

- Multiple classes (**multi-class classification**)

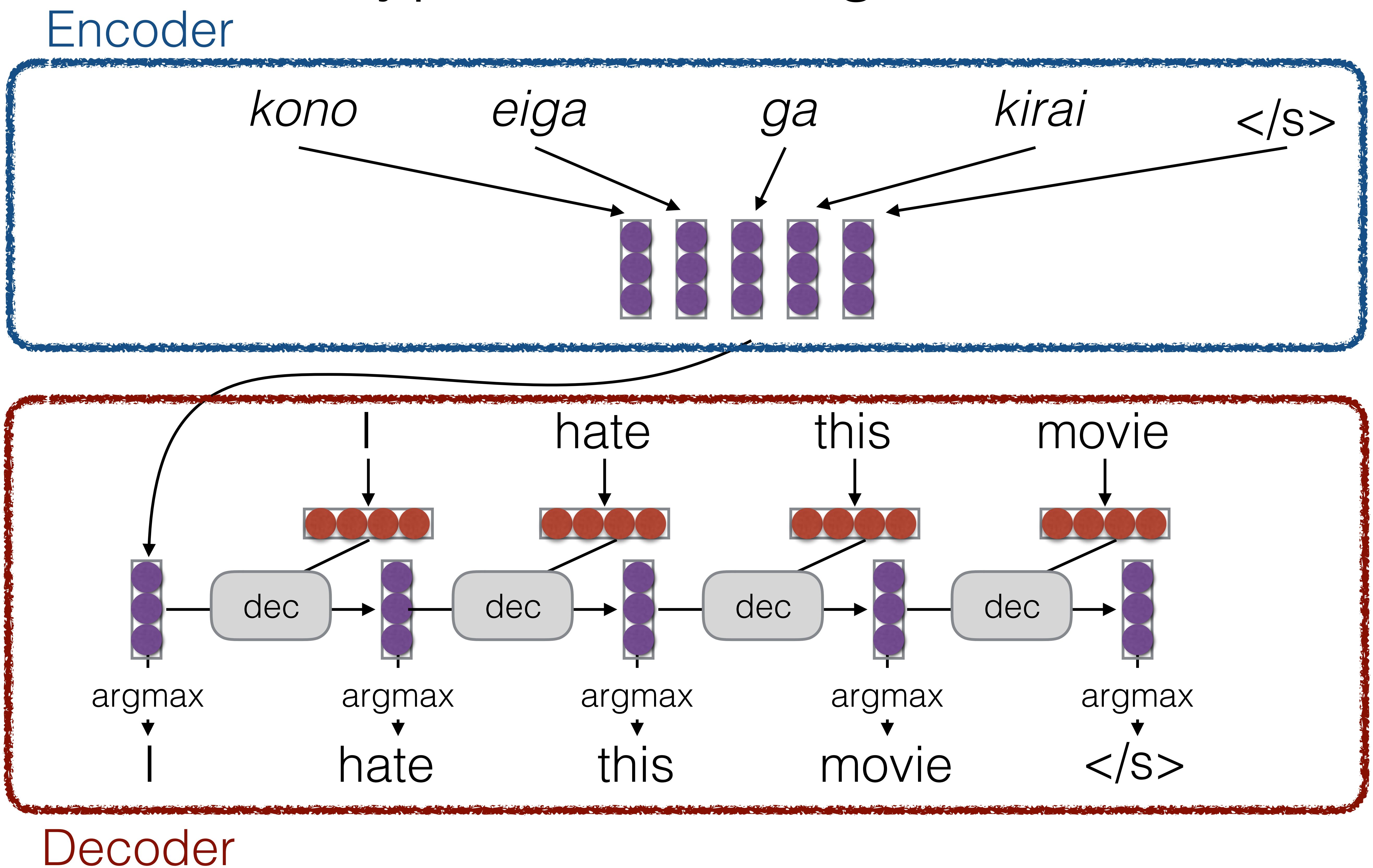
  I hate this movie → very good / good / neutral / bad / very bad

- Exponential/infinite labels (**structured prediction**)

  I hate this movie → PRP VBP DT NN

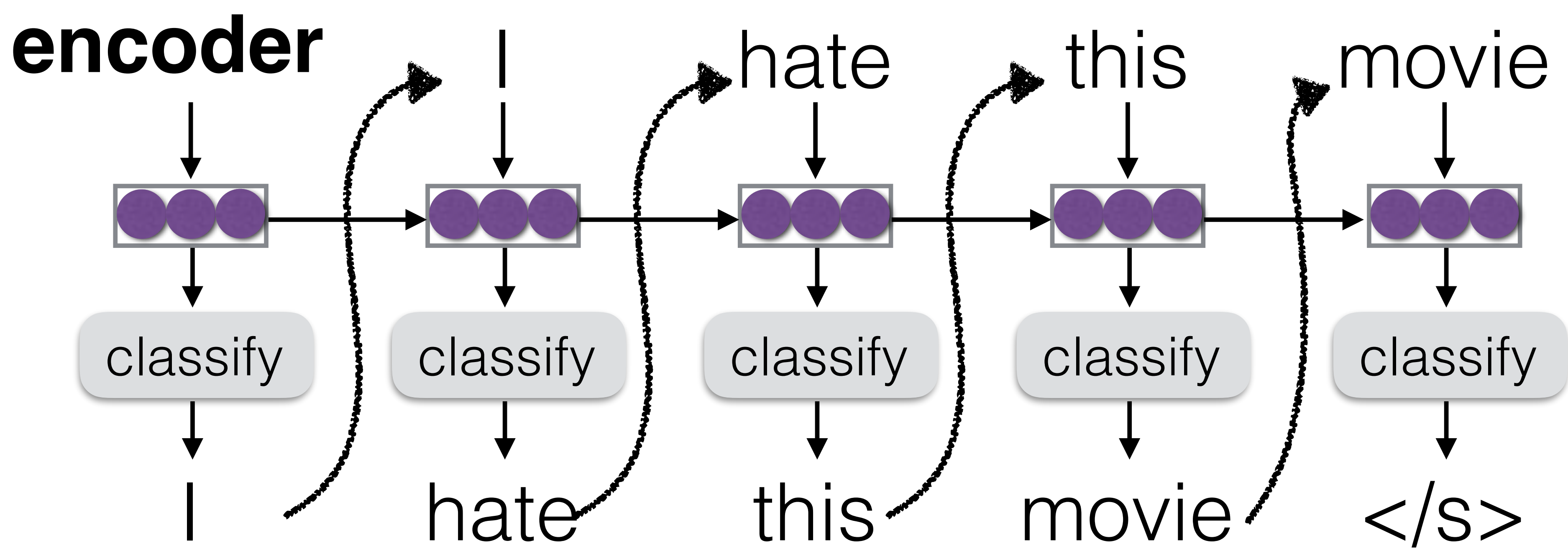
  I hate this movie → *kono eiga ga kirai*

Our Model:
Some Type of Auto-regressive NN



Standard MT System Training/Decoding

Decoder Structure



$$P(E \mid F) = \prod_{t=1}^{T} P(e_t \mid F, e_1, \ldots, e_{t-1})$$

Maximum Likelihood Training

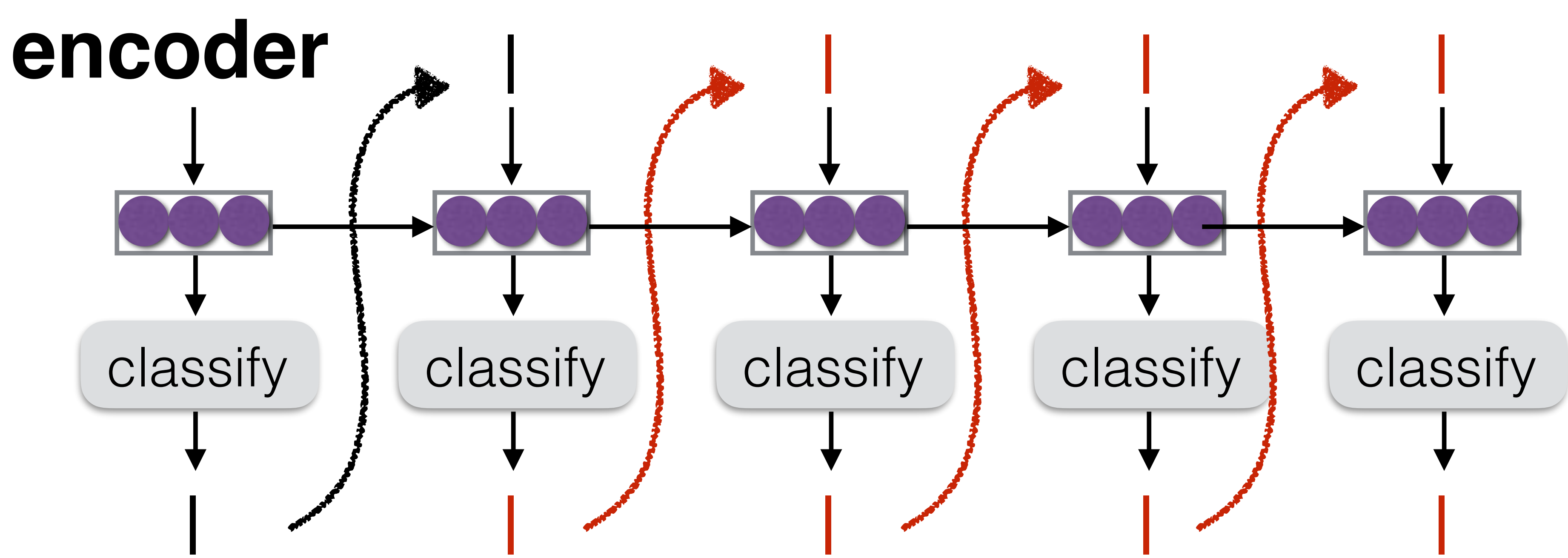
- Maximum the likelihood of predicting the next word in the reference given the previous words

$$\begin{aligned} \ell(E \mid F) &= -\log P(E \mid F) \\ &= -\sum_{t=1}^{T} \log P(e_t \mid F, e_1, \ldots, e_{t-1}) \end{aligned}$$

- Also called "teacher forcing"

Problem 1: Exposure Bias

- Teacher forcing assumes feeding correct previous input, but at test time we may make mistakes that propagate



- **Exposure bias:** The model is not exposed to mistakes during training, and cannot deal with them at test

Problem 2: Disregard to Evaluation Metrics

- In the end, we want good translations
- Good translations can be measured with metrics, e.g. BLEU or METEOR
- Some mistaken predictions hurt more than others, so we'd like to penalize them appropriately

Error and Risk

Error

- Generate a translation

$$\hat{E} = \text{argmax}_{\tilde{E}} P(\tilde{E} \mid F)$$

- Calculate its "badness" (e.g. 1-BLEU, 1-METEOR)

$$\text{error}(E, \hat{E}) = 1 - \text{BLEU}(E, \hat{E})$$

- We would like to minimize error

Problem: Argmax is Non-differentiable

- The argmax function makes discrete zero-one decisions
- The gradient of this function is zero almost everywhere, not-conducive to gradient-based training

Risk

- Risk is defined as the expected error

$$\text{risk}(F, E, \theta) = \sum_{\tilde{E}} P(\tilde{E} \mid F; \theta)\text{error}(E, \tilde{E}).$$

- This is includes the probability in the objective function!
- Differentiable, but the sum is intractable
- Minimum risk training minimizes risk, Shen et al. (2016) do so for NMT

Sampling for Risk

- Create a small sample of sentences (5-50), and calculate risk over that

$$\mathrm{risk}(F, E, S) = \sum_{\tilde{E} \in S} \frac{P(\tilde{E} \mid F)}{Z} \mathrm{error}(E, \hat{E})$$

- Samples can be created using random sampling or n-best search

- If random sampling, make sure to deduplicate

Adding Temperature

$$\mathrm{risk}(F, E, \theta, \tau, S) = \sum_{\tilde{E} \in S} \frac{P(\tilde{E} \mid F; \theta)^{1/\tau}}{Z} \mathrm{error}(E, \hat{E})$$

- Temperature helps adjust for the fact that we're only getting a small sample

Reinforcement Learning

Supervised Learning

- We are given the correct decisions

$$\ell_{\mathrm{super}}(Y, X) = -\log P(Y \mid X)$$

- In the context of reinforcement learning, this is also called "imitation learning," imitating a teacher (although imitation learning is more general)

Self Training

- Sample or argmax according to the current model

$$\hat{Y} \sim P(Y \mid X) \quad \text{or} \quad \hat{Y} = \text{argmax}_Y P(Y \mid X)$$

- Use this sample (or samples) to maximize likelihood

$$\ell_{\text{self}}(X) = -\log P(\hat{Y} \mid X)$$

- No correct answer needed! But is this a good idea?

- One successful alternative: co-training, only use sentences where multiple models agree (Blum and Mitchell 1998)

Policy Gradient/REINFORCE

- Add a term that scales the loss by the reward

$$\ell_{\mathrm{self}}(X) = -R(\hat{Y}, Y) \log P(\hat{Y} \mid X)$$

- Outputs that get a bigger reward will get a higher weight
- Can show this converges to minimum-risk solution
- Quiz: Under what conditions is this equal to MLE?

Credit Assignment for Rewards

- How do we know which action led to the reward?

- Best scenario, immediate reward:

a_1	a_2	a_3	a_4	a_5	a_6
0	+1	0	-0.5	+1	+1.5

- Worst scenario, only at end of roll-out:

a_1	a_2	a_3	a_4	a_5	a_6	
						+3

- Often assign decaying rewards for future events to take into account the time delay between action and reward

Stabilizing MRT/ Reinforcement Learning

Problems w/ MRT/ Reinforcement Learning

- Sampling-based methods tend to be unstable
- It is particularly unstable when using bigger output spaces (e.g. words of a vocabulary)
- A number of strategies can be used to stabilize

Adding a Baseline

- Basic idea: we have expectations about our reward for a particular sentence

	Reward	Baseline	B-R
"This is an easy sentence"	0.8	0.95	-0.15
"Buffalo Buffalo Buffalo"	0.3	0.1	0.2

- We can instead weight our likelihood by B-R to reflect when we did **better or worse than expected**

$$\ell_{\text{baseline}}(X) = -(R(\hat{Y}, Y) - B(\hat{Y})) \log P(\hat{Y} \mid X)$$

- (Be careful to not backprop through the baseline)

Calculating Baselines

- Choice of a baseline is arbitrary
- Option 1: predict final reward using linear from current state (e.g. Ranzato et al. 2016)
  - **Sentence-level:** one baseline per sentence
  - **Decoder state level:** one baseline per output action
- Option 2: use the mean of the rewards in the batch as the baseline (e.g. Dayan 1990)

Increasing Batch Size

- Because each sample will be high variance, we can sample many different examples before performing update

- We can increase the number of examples (roll-outs) done before an update to stabilize

- We can also save previous roll-outs and re-use them when we update parameters (experience replay, Lin 1993)

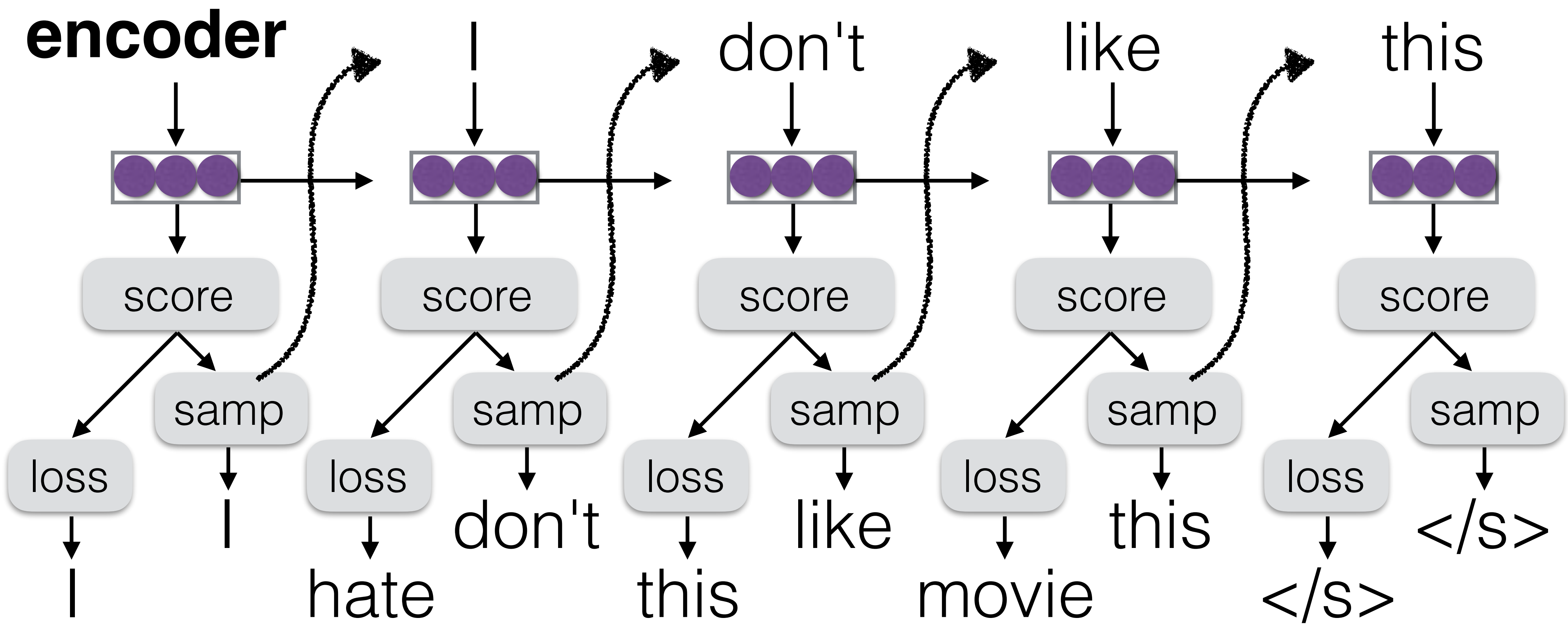
Warm-start

- Start training with maximum likelihood, then switch over to REINFORCE
- Works only in the scenarios where we can run MLE (not latent variables or standard RL settings)
- MIXER (Ranzato et al. 2016) gradually transitions from MLE to the full objective

Corruption-based Approximations

Solution 1: Sample Mistakes in Training (Ross et al. 2010)

- DAgger, also known as “scheduled sampling”, etc., randomly samples wrong decisions and feeds them in

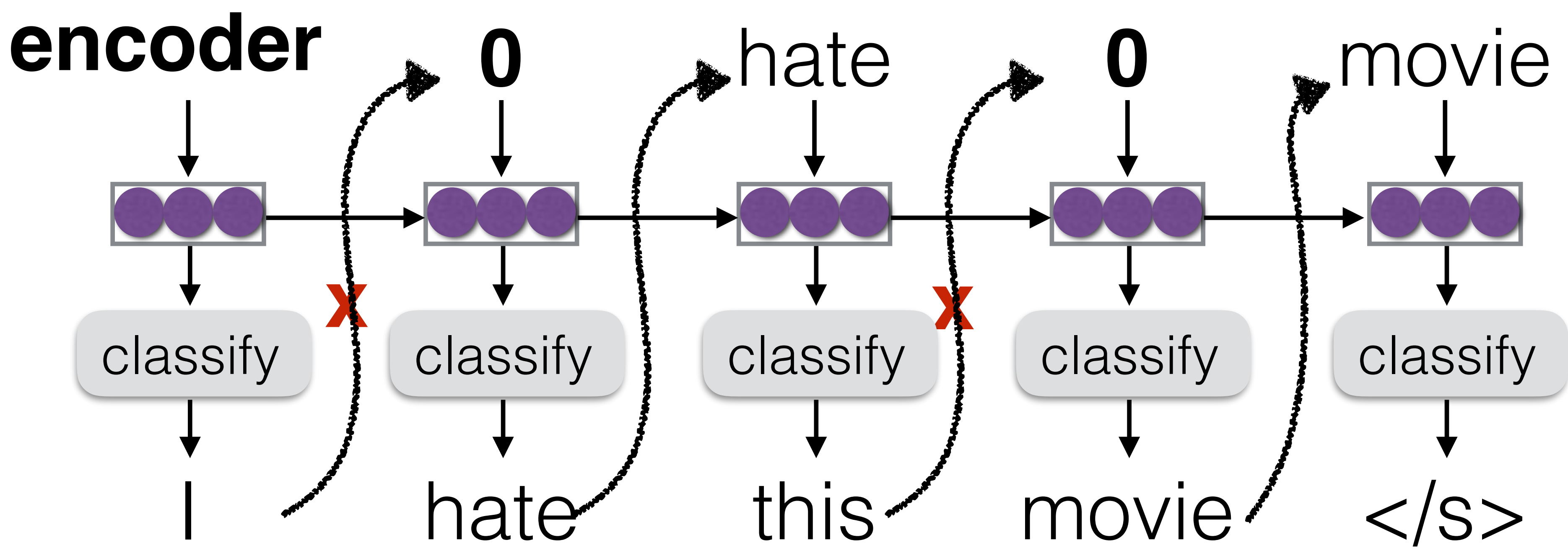


  - Start with no mistakes, and then gradually introduce them using annealing

- How to choose the next tag? Use the gold standard, or create a “dynamic oracle” (e.g. Goldberg and Nivre 2013)

Solution 2: Drop Out Inputs

- **Basic idea:** Simply don't input the previous decision sometimes during training (Gal and Ghahramani 2015)



- Helps ensure that the model doesn't rely too heavily on predictions, while still using them

Solution 3: RAML (Nourozi et al. 2016)

- Reward augmented maximum likelihood
- **Basic idea:** randomly sample incorrect training data, train w/ maximum likelihood

I	hate	this	movie
sample ↓		↕ MLE	
I	**blue**	this	movie

- Exponentiated payoff distribution: sample proportional to goodness of output

$$q(y' \mid y; \tau) \propto e^{r(y', y)/\tau}$$

- Can be shown to approximately minimize risk with entropy regularization

Bonus: SwitchOut (Wang et al 2018)

- Apply RAML-like sampling to source and target side

$$\mathbf{q}^*(\widehat{x}, \widehat{y}|x, y) = \frac{\exp\{s(\widehat{x}, \widehat{y}; x, y)/\tau\}}{\sum_{\widehat{x}', \widehat{y}'} \exp\{s(\widehat{x}', \widehat{y}'; x, y)/\tau\}}$$

- Gives a probabilistic description of data augmentation algorithms for MT

- Good results on WMT en-de, de-en, en-vi

Other Options

Other Options

- **Beam search optimization** (Wiseman and Rush 2016): Try to prevent good hypotheses from falling off the beam

- **Differentiable beam search** (Goyal et al. 2018): turn operations in beam search into differentiable approximations

- **Actor-critic algorithms** (Bahdanau et al. 2016): Create a "critic" that predicts future reward

Questions?

References:

Optimization for Statistical Machine Translation, a Survey (Neubig and Watanabe 2016)

Machine Translation and Sequence-to-sequence Models, Parameter Optimization http://phontron.com/class/mtandseq2seq2018/schedule/optimization.html