Generative Adversarial Network in Neural Machine Translation
Report 1

NGO HO Anh Khoa

A. Problem

The common approach of training RNN is maximizing the log predictive likelihood of each true token in the training sequence known the previous observed tokens. However, in the inference stage, the prediction is based on the previous predicted tokens. This discrepancy between training and inference causes the exposure bias in the inference stage.

B. Solutions proposed

a. Scheduled Sampling SS [Bengio et al]

- Main idea: The training process is modified by using the predicted previous token, forcing the model to deal with its own mistakes.
- Detail: For every token to predict, the model selects the true previous token or predicted token by flipping a coin with a probability. This probability is followed a schedule of decrease depending on the expected speed of convergence.
- Result: [Bengio et al] shows that their approach has a slight improvement in general. [Huszar] demonstrates that it is improper and leads to an inconsistent learning algorithm. In detail, the model, under a trivial solution of SS, memories distribution of tokens conditioned on their positions in the sequence, rather than on the previous tokens.

b. Loss function on the entire generated sequence

- Main idea: The back-propagation is based on the loss on the entire generated sequence (BLUE score) instead of each transition.
- Result: The loss of a specific task is not directly available to be an accurate score of sequence generation for other kind of application.

c. Generative Adversarial Networks GAN

- Main idea:

There are two main parts including a generative and discriminative model. The Generator generates a sequence whereas the Discriminator evaluates this generated sequence. The back-propagation of Generator is based on the feedback of Discriminator.

- Problem:
  - Continuous space of GAN:

    GAN is designed only for the continuous data, which means that it has difficulties in directly generating sequences of discrete tokens. In other words, the gradient of the output of the discriminator shows how to slightly change the data. However, if the output is a word - discrete token, it is impossible to slightly modify this word. There are two approaches: Continuous Space or Reinforcement Learning.
  - Loss for a partially generated sequence:

    GAN gives only the loss/score for an entire sequence. It is non-trivial to balance the quality of a partially generated sequence in the current time and of a full sequence in the future.

d. Professor Forcing PF

- Main idea: It is is based on Adversarial Domain Adaptation. between hidden states from sampling mode (Inference) and teacher forcing mode (Training). There are two components (Generator and Discriminator), two running modes (Teacher-Forcing mode and Free-Running mode).

  Teacher-Forcing mode means that the prediction of the next token is based on the true token. Free-Running mode means that the prediction is based on the self-generated token.

  The Generator generates the behavior of the network (Output and Hidden states) which is indistinguishable whether its inputs from Teacher-Forcing mode) or from Free-Running mode. The Discriminator estimates the probability of the behavior of the network produced in Teacher-Forcing mode.

C. Generative Adversarial Networks GAN and some approaches:

a. An approach of continuous space: [Bowman et al]

This approach replaces the encoder-decoder in translation model by the Variational Autoencoder VAE. In this case, the inputless decoder of VAE [Bowman et al] is a differentiable generator, which could adapt the slightly change from the discriminator. [Bowman et al] shows that the inputless decoder is weaker than standard decoder.

- The expectation of VAE [Bowman et al]:

$$L(\theta, x) = -KL(q_\theta(z,x) \,||\, p(z)) + E_{q_\theta(z|x)}[log\, p_\theta(x|z)] < log\, p(x)$$

  - *KL: How closely the latent variables match a unit gaussian.*
  - *z: Hidden codes*
  - $q_\theta(z,x)$ *: A learned posterior recognition model*
  - $p(z)$*: Prior distribution*

KL shows the distance between a recognition model $q_\theta(z,x)$ and the prior distribution $p(z)$. The Gaussian prior $q_\theta(z,x)$ is the regularizer on the hidden code.

[Hanin et al] uses Domain H-Divergence which relies on the capacity of the hypothesis class (Discrete or Continuous) to distinguish between examples from source or target domain.

In general, for generating text, the problem of this approach (Variational Auto-encoder framework) is that the posterior of the hidden variables would not cover the hidden space, causing difficulties to randomly produce sentences.

b. An approach of domain adaptation with Reinforcement Learning RL:

[He et al] in Dual Learning uses the Language Model to evaluate the quality of a sentence. In the case of GAN, a Discriminator detects whether the input sentence is generated by human or machine.

- Expectation of [He et al]:

$$E[r] = \frac{1}{K}\sum_{k=1}^{K}[r_k|TM_{AB}]$$

  - *Language-model reward $r_{1,k} = LM_B(s_{mid,k})$*
  - *Communication reward $r_{2,k} = \log P(s|s_{mid,k}, TM_{AB})$*
  - *Total reward $r_k = \alpha r_{1,k} + (1-\alpha) r_{2,k}$*
  - *K output sentences*
  - $\nabla_{TM_{AB}}\hat{E}[r] = \frac{1}{K}\sum_{k=1}^{K}[r_k \nabla_{TM_{AB}} log\ P(s_{mid,k}|s,\ TM_{AB})\ ]$

The expectation of [He et al] is the expectation average of K translated sentences with Beam search.

In [Hanin et al], the discriminator returns the Loss of Domain Classifier (Source/Target Domain or Human/Machine-generated sentence). This loss includes the loss of both human and machine generated sentences.

- Expectation of DANN [Hanin et al] = Loss of Label predictor - Loss of Domain classifier.

$$E(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_y^i(\theta_f, \theta_y) - \lambda\left(\frac{1}{n}\sum_{i=1}^{n} L_d^i(\theta_f, \theta_d) + \frac{1}{n'}\sum_{i=n+1}^{N} L_d^i(\theta_f, \theta_d)\right)$$

  - *N examples: $\{(x_i, y_i)\}_{i=1}^{n}$ from Machine domain and $\{x_i\}_{i=n+1}^{N}$ from Human domain.*
  - *Loss of Label predictor: (Input x, True label y)*
    $$L_y(\theta_f, \theta_y) = L_y(\ G_y\ (\ G_f(x, \theta_f)\ ,\ \theta_y),\ y)$$
  - *Loss of Domain classifier: (Input x, True domain d)*
    $$L_d(\theta_f, \theta_d) = L_d(\ G_d\ (\ G_f(x, \theta_f)\ ,\ \theta_d),\ d)$$
  - *Feature extractor G_f : Map an example to new D-dimensional representation. G_f: X -> R^D*
  - *Label predictor G_y : Predict the label of an example. L labels. G_y: R^D -> $[0,1]^L$*
  - *Domain classifier or regressor G_d: Logistic regressor (Shallow NN) G_d: R^D -> [0,1]*

[Li et al], in a similar approach, proposes the reward of RL by using only the probability of sentence being human-generated sentence. Note that based on

reinforcement learning, [Li et al], [Yang et al], [Yu et al] uses the probability instead of log probability [He et al].

- Expectation of [Li et al]:

$$J(\theta) = E_{y\sim p(y|x)}(Q_+(\{x,y\})|\theta)$$

  - $Q_+(\{x,y\})$ *: Probability of human-generated dialogue*
  - $\Delta J(\theta) \approx [(Q_+(\{x,y\}) - b(\{x,y\})]\ \Delta log\pi(y|x)$
  - $b(x, y)$ *: The baseline value to reduce the variance of the estimate while keeping it unbiased.*
  - *x: Input - dialogue history*
  - *y: Response.*

The policy gradient training of [Yang et al] is based on [Yu et al] but the reward is probability of sentence being human-generated sentence [Li et al]. In contrast, for [Yu et al], the discriminator returns the probability of a sentence coming from ground truth data.

- Expectation of [Yang et al]:

$$J(\theta) = \sum_{y_t} G_\theta(y_t|y_{1:t-1}, x).\ R_D^{G_\theta}((y_{1:t-1}, x), y_t)$$

  - $R_D^{G_\theta}$ *: Action-value function of a target language sentence given the source sentence i.e. the expected accumulative reward starting from the state* $(y_{1:t-1}, x)$*, taking action* y_t *and following the policy* G_θ
  - $R_D^{G_\theta}((y_{1:t-1}, x), y_t) = D(x, y_{1:T}) - b(x, y_{1:T})$
  - $D(x, y_{1:T})$*: Reward value for a finished target sequence. Probability of being human-generated sequence.*
  - $b(x, y_{1:T})$ *: The baseline value to reduce the variance of the estimate while keeping it unbiased. In this case, b is a constant.*

Human-generated sequence: The approach of DANN [Hanin et al] proposes a jointly training in which the gradient update of generator is the sum of loss of label predictor and domain classifier with gradient reversal layer. However, [Li et al], [Yang et al] and [Yu et al] have two steps of gradient update of generator. Model updates Generator using reward from Discriminator and then using the human-generated sentence with Teacher-Forcing (Only if the reward is larger than the baseline value [Li et al]). The number of gradient update iteration is also a hyper-parameter.

Reward for a sentence: The reward of [Yu et al] is keyword retrieval confidence. The manually defined reward function can not possibly cover all crucial aspects and can lead to suboptimal generated sequence. In fact, the optimization target of discriminator [Yu et al] is minimizing the cross entropy between the ground truth label and the predicted probability. In a different approach [Li et al] and [Yang et

al], the optimization target is maximizing the difference between human-generated and machine-generated sentence.

c. Two approaches for partially generated sequence:

[Li et al] mentioned that Discriminator Strategy DS is more time-effective but less accurate than Monte Carlo search MC. In fact, MC requires repeating the sampling process of each prefix of each sequence, which is significantly time-consuming.

■ Monte Carlo search [Yang et al] and [Yu et al]

This approach tackles the loss for a partially generated sequence by generating the full sentence based on the known part of this sentence.

- Reward for a partially generated sequence:

$$R_D^{G_\theta}((y_{1:t-1}, x), y_t) =$$

- $\frac{1}{N}\sum_{n=1}^{N} D(x, y^n_{1:L_n}) - b(x, y^n_{1:L_n}),\ y^n_{1:L_n} \in MC^{G_\theta}((y_{1:t}, x), N)$ for t < L . For partially generated sequence.
- $D(x, y_{1:T}) - b(x,\ y_{1:T})$ for t = L. For full sequence of size L.

  - $R_D^{G_\theta}$ *: Action-value function of a target language sentence given the source sentence i.e. the expected accumulative reward starting from the state* $(y_{1:t-1}, x)$*, taking action* y_t *and following the policy* G_θ
  - $D(x, y_{1:T})$ *: Reward value for a finished target sequence*
  - $b(x,\ y_{1:T})$ *: The baseline value to reduce the variance of the estimate while keeping it unbiased.*
  - $MC^{G_\theta}((y_{1:t}, x), N)$ *generates N samples of the rest tokens of a sequence.*
  - L_N *: Length of the sentence sampled by the i'th Monte Carlo search*
  - $(y_{1:t}, x)$ *: Current state*
  - $y^N_{1:L_N}$ *: Sampled based on the policy G*
  - $\{y^1_{1:L_1}, \dots, y^N_{1:L_N}\} = MC^{G_\theta}((y_{1:t}, x), N)$

■ Discriminator Strategy [Li et al]

Discriminator assigns the reward to both both full and partial sequence.

- A generated sequence is broken into partial sequences including human-generated tokens and machine-generated tokens.

- One example is randomly sampled from machine-generated and human-generated tokens for the negative and positive example respectively.
- $\Delta E(\theta) \approx \sum_t (Q_+(\{x, Y_t\}) - b(\{x, Y_t\}) \Delta logp(y_t|x, Y_{1:t-1})$

  where each partially-generated sequence $Y_t = y_{1:t}$
  - *$Q_+(\{x,y\})$: Probability of human-generation dialogue*
  - *$b(\{x,y\})$: Baseline value to reduce the variance of the estimate while keeping it unbiased*
  - *x: Input - dialogue history*
  - *y: Response.*

Baseline value: Baseline value is to reduce the variance of the estimate while keeping it unbiased. [Li et al] trains another neural network model to estimate the value (future reward) of current state under the current policy. [Yang et al] sets it as a constant. [Yu et al] does not mention about this value.

d. Problem of Strong Discriminator

For GAN, the generator needs the reward passed back from the discriminator. This reward guides the generator to reach the gold-standard target. If in the beginning of the training period, the discriminator is strong, the generator could receive a reward insignificant, leading to a breakdown of the training process.

● Teacher Forcing:

[Li et al] uses a similar approach of Teacher Forcing. In this case, for each training iteration, there are two periods of update for discriminator and generator. For the period of generator update, the generator updates with the reward from the discriminator and the reward of the target sentence (Teacher Forcing). In detail, the discriminator automatically assign a reward of 1 to the golden target-language sentence and the generator uses this reward to update itself.

The number of updates iteration is also a solution to keep balance between Generator and Discriminator.

Moreover, the accuracy of Discriminator is an hyperparameter to decide whether to back-propagate from Discriminator into Generator and to update Discriminator. In Professor Forcing [Lamb et al], if the discriminator accuracy is greater than 75%, model back-propagates from Discriminator into Generator. If its accuracy is greater than 99%, model does not update Discriminator.

● CNN and RNNs for Discriminator

[Yang et al] uses also Teacher Forcing, a same approach of [Li et al]. In addition, [Yang et al], based on the result of GAN with RNN(BiLSTM, LSTM) and CNN, shows that RNN-based discriminator is not stable because of its sharp decrease of BLEU score. They explained that this discriminator RNNs could achieve

the high score after few updates, which means that it is too strong for the generator. Therefore, they uses CNN for their experiment. They modify also the initial accuracy of the discriminator.

For the comparison of CNN and RNN, [Yin et al] proposes an experiment between CNN, GRU and LSTM. RNNs are well suited to encoder order information and long-range context dependency. CNN are considered good at extracting local and position-invariant features, which is good for classification. In detail, RNN (Bi-RNN) chooses the last hidden state to represent the sentence, which probably causes the wrong text classification prediction because this classification mostly depends on a few local regions. GRU and CNN are comparable when sentence length is small. [Yin et al] concludes that the performance in text classification depends on how often the comprehension of global/long-range semantics is required.

CNN Discriminator has a convolutional layer and a max-pooling operation. An input sentence is a matrix (word dimension * sentence length). The convolution layer involves a filter of n-gram, in which each filter creates a one feature map at every position in the sentence. The model applies a-max-over-time pooling operation to the feature map and take its maximum value. In this case, the pooling operator tries to capture the most important feature (Maximum value).

The discrimination between Human and Machine-generated requires the comprehension of long-range semantics. Therefore, RNN is too good for this task. In our case, RNNs should be used for generator (translation model) and CNN for discriminator.

- Other strategies

In the similar case, for Dual Learning, [He et al] proposes the soft-land strategy, which improves the capacity of the generator. The soft-landing strategy means that each minibatch has half sentence from monolingual data and half sentence from bilingual data in the beginning of training procedure. The percentage of monolingual sentences gradually increases. The objective is maximizing the weighted sum of the reward (Monolingual data) and of the likelihood (Bilingual data).

e. Professor Forcing, Sequence GAN, Domain Adversarial NN and Scheduled Sampling

The models such as Professor Forcing (PF), Sequence GAN (SeqGAN), Domain Adversarial NN (DANN) are based on the framework of GAN by using two models Generator and Discriminator model.

In this group, the main difference between PF and other GANs is the objective of discrimination. In fact, SeqGAN discriminates between human-generated samples (Real) and machine-generated samples (Model-generated). DANN is the case of discrepancy between source and target domain. For PF, the model concentrates on the input of the token prediction. The input is from the training set or the self-generated token.

On point important is that PF and DANN use the intermediate hidden vectors of the generator for the discrimination procedure, which makes the system differentiable. SeqGAN uses its sequence output.

An advantage of PF over GANs is that PF can be used to learn a generative model over discrete random variables without requiring to approximate backpropagation through discrete spaces.

In comparison between GANs and SS, SS does not require to train an additional discriminator network.

D. Models

a. Language Model or Discriminator in the role of evaluating the translated sentence (Not BLUE score): The input and output sentence have a fixed length T [Yang et al].

- The reward of LM [He et al]: The log-likelihood of the translated sentence

$$r_{LM} = \prod_{l=1}^{T} P(y_l | y_{<l-1}, x_{1:l})$$

- The reward of Discriminator: The probability of a sentence being human-generated sentence [Yang et al], [Li et al] and Baseline.

$$r_D = Q_+(y_{1:T}) + b(y_{1:T}, x_{1:T})$$

- Reward in Monte Carlo search or Discriminator strategy for Partially generated sentence.
- The gradient of translation model in these cases above:

$$\nabla_{TM} = [r \nabla_{TM} \sum_{l}^{T} logP(y|x_{1:T}, y_{<l-1})]$$

- Translation model creates K translated sentences with Beam search:

$$\nabla_{TM} = \frac{1}{K} \sum_{k=1}^{K} [r \nabla_{TM} \sum_{l=1}^{T} logP(y|x_{1:T}, y_{<l-1})]$$

b. Teacher Forcing in comparison with the true translated sentence.

c. Variational AutoEncoder or Tradition Encoder-Decoder in the role of Generator (Translation Model).

E. Reference:

a. [Huszar]: How (not) to train your generative model: Scheduled sampling, likelihood, adversary?

b. [Bengio et al]: Scheduled Sampling for Sequence Prediction with Recurrent Neural Network

c. [He et al]: Dual Learning for Machine Translation

d. [Bowman et al]: Generating Sentences from a Continuous Space

e. [Hanin et al]: Domain-Adversarial Training of Neural Networks

f. [Yang et al]: Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets

g. [Yu et al]: SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient
h. [Li et al]: Adversarial Learning for Neural Dialogue Generation
i. [Yin et al]: Comparative Study of CNN and RNN for Natural Language Processing
j. [Lamb et al]: Professor Forcing: A New Algorithm for Training Recurrent Networks