10-708: Probabilistic Graphical Models, Spring 2020

15:Case Study of Deep Generative Models: Text Generation

Lecturer: Eric P. Xing *Scribe: hel1, mingjie1, yukaih2, wenhaod*

1 Text Generation Tasks and Center Goals

The goal of text generation is to generate natural language from input data or machine representations. It spans a broad set of natural language processing (NLP) tasks. These tasks includes dialog system, machine translation, summarization, description generation, captioning and speech recognition.

There are basically two center goals in all these tasks. The first center goal is generating human-like, grammatical, and readable text, or so called natural language. The second goal is generating text that contains desired information inferred from inputs. For example, in machine translation the generated target sentence should has the same meaning with the source input sentence. In data description tasks, the generated report should describe the input data table. And for other tasks like attribute control the generated sentence should also contain the same kind of information as input like generating "I like this restaurant" for a positive sentiment input. Same goals is also in conversation control that we should control conversation strategy and topic to the specified input.

2 Common Model for Text Generation

We will first take a look at the first center goal: how to generate natural language.

2.1 Language Model

One of the most basic model for text generation tasks is language model. In language model, the sentence $\mathbf{y}$ is modeled as a series of tokens y_t, and the probability of the sentence $p_\theta(\mathbf{y})$ is computed by decomposing the probability of the whole sentence across times steps, or tokens. And the probability of each token y_t is modeled by a conditioned probability to its previous sequence$\mathbf{y}_{1:t-1}$. The expression is as below:

$$\mathbf{y} = y(y_1, y_2, ..., y_T) \tag{1}$$

$$p_\theta(\mathbf{y}) = \prod_t p_\theta(y_t|\mathbf{y}_{1:t-1}) \tag{2}$$

To implement this model, we can use recurrent neural networks like LSTM.

2.2 Conditional Language Model

In conditional language model, we can specify the context x to incorporate the input data . For example, in machine translation, x is the input sentence. During the inference, the probability is conditioned on input

x. Expression is as below:

$$\mathbf{y} = y(y_1, y_2, ..., y_T) \tag{3}$$

$$p_\theta(\mathbf{y}|\mathbf{x}) = \prod_t p_\theta(y_t|\mathbf{y}_{1:t-1}, \mathbf{x}) \tag{4}$$

3 Common Learning Algorithm

3.1 Maximum Likelihood Estimation (MLE)

One of the most popular and simplest training method is Maximum Likelihood Estimation (MLE). In MLE, we will maximize the data log-likelihood $L(\theta)$ from the given ground truth data $\mathbf{y}^*$. The expression for training is as below:

$$\mathbf{y}^* = (y_1^*, y_2^*, ..., y_T^*) \tag{5}$$

$$\theta = \arg\max_\theta L(\theta) = \arg\max_\theta p_\theta(\mathbf{y}^*|x) = \arg\max_\theta (\log \prod_t p_\theta(y_t^*|\mathbf{y}_{1:t-1}^*, \mathbf{x})) \tag{6}$$

For the evaluation, we have different metrics for different tasks to evaluate similarity between the output and ground truth. For example, BLEU is used for machine translation and ROUGE is used for summarization.

3.1.1 Two Issues of MLE

There are two issues of MLE. The first issue is called exposure bias [Ranzato et al., 2015]. In the training process, the next token is predicted given the previous ground-truth sequence: $(y_t^*|\mathbf{y}_{1:t-1}^*, \mathbf{x})$. But in evaluation process, the ground truth is not exposed to the model. The next token is predicted given previous sequence that are generated by the model it self: $(\hat{y}_t|\hat{\mathbf{y}}_{1:t-1}, \mathbf{x})$. So if there is an error in the model, the error will affect subsequent tokens. The second issue is the mismatch between training and evaluation criteria. As described in the training and evaluation process, we use log-likelihood for training metrics but use BLUE for evaluation in machine translation.

3.1.2 Possible Solutions

Reinforcement learning [e.g., Ranzato et al., 2015] is one of the solutions to deal with the mismatch between training and evaluation criteria. In reinforcement learning, a reward function $R(\mathbf{y}, \mathbf{y}^*)$ is defined on sequence $\mathbf{y}$ and ground truth $\mathbf{y}^*$. And the training process is to maximize the expected reward. For example, if we use BLEU as reward function R for translation tasks, then we will have the same metrics for training and evaluation. Expression is as below:

$$\max_\theta \mathbb{E}_{p_\theta(\mathbf{y})}[R(\mathbf{y}, \mathbf{y}^*)] \tag{7}$$

But there are also problems for reinforcement learning. For example, the sequence space is extremely large (50000^{50} for a sentence with length of 50 and vocabulary of 50000). So there will be high variance and poor exploration efficiency during training process. This means most of the time, the model may generate something not very meaningful.

There are also many recent works to make training more practical. For example, Reward Augmented Maximum Likelihood (RAML) [Norouzi et al.,16] adds reward-aware perturbation to the MLE data examples to close the gap between training and evaluation criteria. RAML can also let the model see mistakes in the data rather than just ground truth data. In Softmax Policy Gradient (SPG) [Ding & Soricut, 17], reward

distribution is used for effective sampling and estimating policy gradient. There are also other algorithms like Data noising [Xie et al.,17] that adds random noise to data so that model is exposed to noise and mistake during training time.

All these algorithms are special instances of a generalized entropy regularized policy optimization (ERPO) framework. The differences are in the choice of rewards and the values of hyperparameters α and β.

4 Generalized Entropy Regularized Policy Optimization (ERPO)

4.1 General Framework

The objective of the generalized ERPO is:

$$L(q, \theta) = \mathbb{E}_q[R(\mathbf{y}|\mathbf{y}^*)] - \alpha KL(q(\mathbf{y}|\mathbf{x})||p_\theta(\mathbf{y}|\mathbf{x})) + \beta H(q) \tag{8}$$

where $p_\theta(\mathbf{y}|\mathbf{x})$ is the sequence generation model, $R(\mathbf{y}|\mathbf{y}^*)$ is the reward function and $q(\mathbf{y}|\mathbf{x})$ is the variational distribution. Therefore, in ERPO, the objective is to maximize the reward function under q, minimize the KL divergence between q and the model $p\theta$, and regularized by the entropy of q.

The objective can be solved with an EM-style procedure.

In the E-step:

$$q^{n+1}(\mathbf{y}|\mathbf{x}) \propto \exp(\frac{\alpha \log p_\theta^n(\mathbf{y}|\mathbf{x}) + R(\mathbf{y}|\mathbf{y}^*)}{\alpha + \beta}) \tag{9}$$

In the M-step:

$$\theta^{n+1} = \text{argmax}_\theta \mathbb{E}_{q^{n+1}}[\log p_\theta(\mathbf{y}|\mathbf{x})] \tag{10}$$

α and β have implications for the algorithm. As $\alpha \to \infty$, $q^{n+1} = p_\theta^n$ and the objective corresponds to minimizing KL divergence. As $\beta \to \infty$, q^{n+1} becomes a uniform distribution and the objective corresponds to maximizing the entropy of q.

4.2 MLE

MLE can be interpreted as a ERPO. Specifically, if the reward function is:

$$R = R_\delta(\mathbf{y}|\mathbf{y}^*) := \begin{cases} 1 & \text{if} \quad \mathbf{y} = \mathbf{y}^* \\ -\infty & \text{otherwise} \end{cases}$$

and $\alpha \to \infty$, $\beta = 1$.

Then the E-step becomes:

$$q(\mathbf{y}|\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \mathbf{y} = \mathbf{y}^* \\ 0 & \text{otherwise} \end{cases}$$

and the M-step becomes:

$$\theta^{n+1} = \text{argmax}_\theta \log p_\theta(\mathbf{y}^*|\mathbf{x})$$

Therefore, MLE can be viewed as a policy optimization with a δ-function reward. In this formulation, any exploration beyond the training data will not be exposed to the model and this will result in exposure bias, which is one of the drawback of MLE. On the other hand, due to the restriction, q becomes the empirical distribution, so the implementation of the algorithm is very simple and efficient

4.3 RAML

In RAML [Norouzi et al., 16], the reward function can be a common reward such as the BLEU$(\mathbf{y}|\mathbf{y}^*)$. $\alpha \to 0$ and $\beta = 1$.

In this scenario, the E-step becomes:

$$q(\mathbf{y}|\mathbf{x}) \propto \exp(R(\mathbf{y}, \mathbf{y}^*))$$

and the M-step becomes:

$$\max_\theta \mathbb{E}_q[\log p_\theta(\mathbf{y}|\mathbf{x})]$$

Compared to MLE, RAML use a smoother reward function and it can be exposed to a larger exploration space.

4.4 SPG

Softmax Policy Gradient (SPG) [Ding & Soricut, 17] is another special case of ERPO. The E-step and M-step are the same as previous two methods, the reward function could be a common reward such as $BLEU(\mathbf{y}, \mathbf{y}^*)$. The only difference is that in SPG we set $\alpha = 1$ and $\beta = 0$, which is contrary to the MLE and RAML methods. The final updating equation is:

$$\text{E-step:} \quad q(\mathbf{y}|\mathbf{x}) \propto p_\theta(\mathbf{y}|\mathbf{x}) \exp(R(\mathbf{y}, \mathbf{y}^*))$$

$$\text{M-step:} \quad \max_\theta E_q \left[\log p_\theta(\mathbf{y}|\mathbf{x})\right]$$

SPG uses both the model distribution and the reward for exploration, therefore leading to the largest exploration space. The advantage is that the chance to find the optimal solution is larger, while the disadvantage is that learning difficulty is highly increased. Some training tricks are needed to resolve this difficulty.

4.5 Data Noising

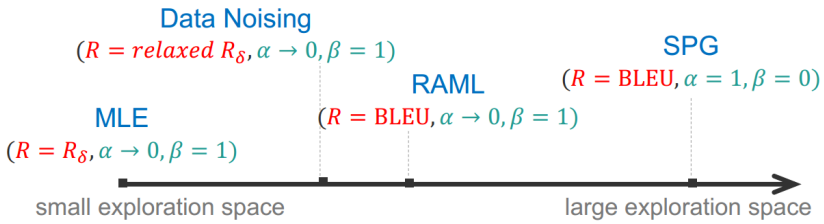
Data Noising is also a special case of ERPO with $\alpha \to 0$ and $\beta = 1$. The difference is that the reward function is a locally relaxed variant of $R_\delta(\mathbf{y}|\mathbf{y}^*)$. For example, we can use this reward function:

$$R'_\delta(\mathbf{y}|\mathbf{y}^*) := \begin{cases} 1 & \text{if} diff(\mathbf{y}, \mathbf{y}^*) = 1 \\ -\infty & \text{otherwise} \end{cases} \tag{11}$$

To implement this reward function, we can randomly replace a single token with another uniformly picked token, which is much easier than adding reward-aware noise in RAML.

5 Interpolation algorithm

After introducing previous four algorithms, we can summarize them in a hyperparameter space. the exploration space can be used to compare these methods, and the relation is shown below:



Every algorithm corresponds to a point in the hyperparameter space and the position is defined by the selection of reward function, α, and β. From left to right, the exploration space increases. There exists a trade-off here: when the exploration space is large, we will have better test performance in theory, but more difficult for training.

An intuitive idea to balance this trade-off is interpolating among the algorithms. We can start from MLE hyperparameter values, and then gradually anneal to the SPG hyperparameter values. This procedure will make the training easy at the beginning and gradually increase the exploration space later.

6 Unsupervised Controlled Generation of Text

Recall that we have two central goals in text generation tasks. In previous sections, we have introduced how to generate human-like, grammatical, and readable text. Next, we will focus on generating text that contains desired information inferred from inputs.

Usually, we need lots of supervision data to achieve the second goal. Though tasks such as machine translation and data description can easily acquire millions of data, for some tasks, we cannot access to these amounts of data. For example, in attribute control task (i.e. modify sentiment from positive to negative) and conversation control task (i.e. control conversation strategy and topic), there is nearly no existing supervision data. Therefore, we have to consider unsupervised controlled generation method.

There are two kinds of generative level in general: sentence-level control and conversation-level control. Two representative works of the sentence-level control are text attribute transfer and text content manipulation, and target-guided open-domain conversation is a typically conversation-level control. In the following part, we will mainly introduce these three tasks.

6.1 Text Attribute Transfer

The task is that, given a sentence, we want to modify this sentence to have a desired attribute value while keeping all other aspects unchanged. Let's say we want to transfer sentiment from negative to positive. Here is an input sentence for example:

"It was super dry and had a weird taste to the entire slice."

we want to modify the sentence to have a positive sentiment like:

"It was super fresh and had a delicious taste to the entire slice."

The goal here is we want to change the content from "dry" to "fresh" and from "weird" to "delicious", but still keep all other aspects unchanged. The application of this kind of tasks can be like the personalized article writing, conversation systems, and authorship obfuscation.

With formal formulation, given an input sentence x and the original attributes a_x. For the target, we have target sentence y and target attribute a_y. Now we want to generate a target sentence y, where y has the desired attributes a_y and keeps all attribute-independent properties of x:

$$Task : (x, a_y) \rightarrow y$$

In the training setting, we only have (x, a_x), but no ((x, a_x), (y, a_y)) for training. So the next is how we can use this data available to train a particular conditional generation model in this case.

The solution is that we design an encoder, which encodes the input sentence as a feature vector z. And we concatenate this feature vector with the attribute, where the value is a_y for the positive sentiment or 0 for the negative sentiment. Then we feed the concatenated input into a decoder to generate y, where the model can be represented as:

$$p_\theta(y|x, a_y)$$

The key intuition for learning is we decompose the task into competitive sub-objectives and use direct supervision for each of the sub-objectives.

The first objective is we want y to keep all attribute-independent properties of x, which means y must be fairly close to the input of x. So a very straightforward and simple way is to enforce the similarity between x and y, where we just use the auto-encoding loss:

$$(x, a_x) \rightarrow x$$

The second objective is we want y has the desired attributes a_y, where we use a classification loss given the produced y sample to a pre-trained sentiment classifier f:

$$\hat{y} \sim p_\theta(y|x, a_y), f(\hat{y}) \rightarrow a_y$$

So there are basically two loss functions and we will optimize these two loss functions jointly.

6.2 Text Content Manipulation

The goal here is that we want to generate a sentence to describe the contents in a given data record like controlling the writing style by using the writing style of a reference sentence.

The method is pretty similar to the one in "Text Attribute Transfer", where we also decompose the task into competitive sub-objectives and use direct supervision for each of the sub-objectives.

6.3 Target-guided Open-domain Conversation

There are basically three sets of conversation systems.

The first set is the Task-oriented dialog, where we use this dialogue system to address the specific tasks like booking a flight or reserving a restaurant. However, the task-oriented dialog is a close domain conversation, which means the conversation system can only do a specific task, not any anything else.

The second set is Open-domain chit-chat conversation. The goal is to improve the user engagement, and the metric is how long this chat bot can keep it conversing with a human, where the conversation is random without any control of topics or other aspects.

The third set is Target-guided conversation, which combines previous two types of conversation. It is still an open-domain conversation that basically can converse with human about whatever topic, but it controls the conversation strategy to reach a desired topic at the end of the conversation at the same time.

There are two goals for target-guided open-domain conversation. The first goal is to reach a desired topic in the end of conversation starting from any topic. The second goal is to achieve a smooth transition in natural conversation.

The challenge is that we have no supervised data for the task. The solution to this problem is using competitive sub-objectives and partial supervision. More specifically, we use rich chit-chat data to learn smooth single-turn transition for generating natural conversation. We can also use rule-based multi-turn planning to reach desired target, that means use keyword of each response to guide the topic closer to the target at each step. The basic step is as follows. At first, we extract keyword from the input sentence. Then we use the learned kernel-based topic transition and target-guided rule to transit to keywords that are close in the word embedding space and make sure the next keywords must get closer to the target keyword. At last, keyword conditional response retrieval is done to generate the conversation.