Variational Autoencoder for Text Modeling

Zhao Liu
Department of Mathematics
The University of Texas at Austin
zliu@math.utexas.edu

Abstract

In this project we implement a CNN-VAE framework for text modeling. To let CNN generate sentences word-by-word, the Decoder part of VAE uses a shifting trick in manipulating the convolution layers. However, due to the limitations of our model, we observe the 'KL collapse' phenomena in loss functions.

1 Introduction

Generative models (decoder) is a major component of natural language understanding. Among them, the Encoder-Decoder type structure is often widely used, such as in machine translation, abstractive text summarization and document modeling. In this project, we focus on a specific instance of this class: the Variational Autoencoder(VAE).

Variational autoencoders (VAE) is introduced by (Kingma and Welling, 2013) for modeling handwritten digits and Face images. In just four years, it have emerged as one of the most popular approaches to unsupervised learning of complicated distributions. Compared to traditional function approximators (which may require strong assumptions about the data and rely on costly inference methods like Markov Chain Monte Carlo(Doersch, 2016)), VAEs require weak assumptions on the model and train Neural Networks with fast back-propagation in SGD.

VAEs have already been widely applied in generation and interpolation of different types of images. However, the application of this new framework has been quite limited. In 2015, Bowman et.al. (Bowman et al., 2015) studied many interesting properties in generation of sentences by standard LSTM-RNN, which is a natural choice in NLP, but they discovered higher perplexity for LSTM-VAE than usual LSTM model. The main difficulty is the collapse of the latent loss (represented by the KL divergence term) to zero. In this case, the generator tends to completely ignore latent representations (the likelihood term) and collapses to a standard language model. Other related works including (Miao et al., 2016) apply VAE to bag-of-words representations of documents, achieving good results in perplexity. (Yang et al., 2017) used Dilated Convolutions for VAE, and (Semeniuta et al., 2017) proposed a hybrid VAE model mixing CNN and RNN.

In this project, we explore VAE for text modeling by replacing LSTM generators with 'shifted' Convolutional Neural Nextwork layers, which shares similarity with (Yang et al., 2017). We also investigate the loss in terms of KL divergence term and the log-likelihood term, which shows similar phenomena as 'KL' collapse, but not . We also tried to apply it to the generation of text in controlled cases, however, the results are not quite positive.

2 The CNN-VAE Algorithm

In this section we briefly explain the VAE framework introduced by (Kingma and Welling, 2013), and describe the structure of

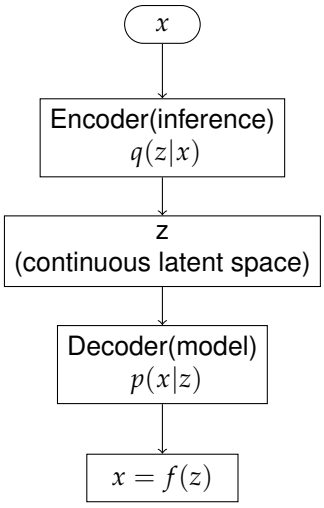

Figure 1: Structure of VAE

our CNN-VAE alogrithm where the Decoder part is a 'shifted' multi-layered CNN and the Encoder part is simply multi-layered CNN with a fully connected layer.

2.1 Variational Autoencoder

The Variational Autoencoder is composed of two parts: the Encoder(inference) part and the Decoder(model) part (See Figure 1). The key point is modeling the distribution of latent variable z, which serves as an internal representation of input text x, by the posterior distribution $q(z|x)$. For convenience of calculating KL-divergence, and using the 'reparameterization trick', typically the prior $p(z)$ is set to be $N(0, I)$, and $q(z|x)$ also a multi-dimensional Gaussian, with location and scale to be learned in the training process.

The total loss function of VAE is defined as

$$J_{vae} = -KL(q(z|x)||p(z)) + \mathbb{E}_{z\sim q}[\log(p(x|z))]$$

The KL-divergence term serves as a regularization. If it is ignored, then the term $\mathbb{E}_{z\sim q}[\log(p(x|z))$ drives the variance of $q(z|x)$ to be small, and the training process collapses as an autoencoder (Yang et al., 2017). The KL-divergence term drives the model away from modeling each x as a single latent variable. However, it has been observed (Bowman et al., 2015) that the KL term vanishes during training, that is, the posterior $q(z|x)$ is driven very fast towards the prior $N(0, I)$. Then it goes to another extreme: the model does not store any information from training data x. It might be related to what type of decoder we use, for example, for RNN-based decoders, the high modeling power with sufficiently small history can achieve low reconstruction errors while not relying on the latent vector provided by the encoder.

Here, we use a 'shifted' CNN as the decoder. The benefit is that we can control the 'effective context width' by controlling the width of convolution kernel, and that it can be tuned to be either largely relied on previous words' history (like LSTM-RNN) or simply like a bag-of-words. We next describe the structure of our decoder.

2.2 Structure of shifted CNN Decoder

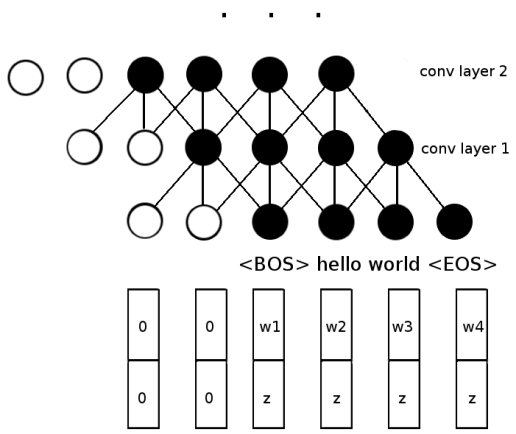

Figure 2: Structure of CNN Decoder

After passing a convolution layer, each generated word is dependent on all words in the input sentence x. We want to avoid this: it is natural to require the t-th predicted word depending only on previous words: $x_{\leq t}$. We made it possible by shifting the input of each layer.

In each one-dim convolution layer, we use a filter of width k, stride 1. Passing each convolution layer, the length of output is the length of input minus $k-1$. So we pre-pad $k-1$ zero vectors in the front, and feed into the next layer. Therefore, as shown in Figure 2, the t-th word in black depends only on $x_{\leq t}$ in the input layer.

We note that this is the simplified version of the Dilated Convolutional Decoders described in (Yang et al., 2017). For simplicity of the model, we did not apply dilation and directly use the idea of shifting.

Further, note that the first input layer is the latent vector z concatenated with the word-embedding vectors of x. At training state, x is the tokenized sentences. At the test state, we simply feed into a random z and '<BOS>' (padded to the maximum sequence length). Each next word is generated by a greedy algorithm maximizing the probability.

2.3 Overall Model Structure

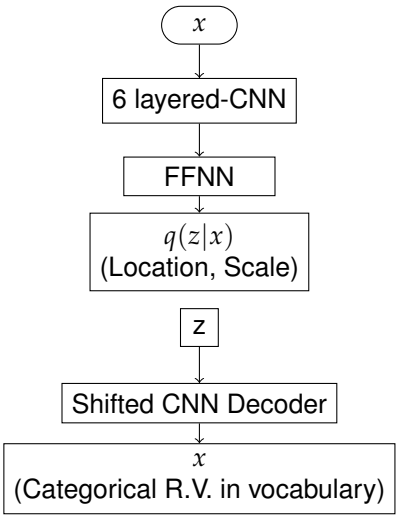

Figure 3: Structure of Model

The overall model structure is described by Figure 2. We also use CNN as the encoder, with same number of layers and same width and stride, but no shifts. Each layer is followed with Relu activation function.

3 Implementation Details

We implement our algorithm with `tensorflow` and `edward` packages.

1. Dataset

We use Yelp review data for restaurants and hotels (`https://www.yelp.com/dataset/download`), where the reviews contain paragraphs as long as 500 words. We set the maximum sentence length to be 25, take the first sentence of each review, and drop those sentences which exceeds length = 25.

2. Word Embeddings

The same Word Embeddings in homework3, with Vocabulary size = 17615, embedding dimension = 50.

One aspect we notice is that, the glove word-embeddings does not contain <BOS> and <EOS> symbols. We use '.' as <EOS>, and set <BOS> as a tensorflow variable, learning it during the training process.

Also, in our homework3 codes, 'UNK' is added into the word-embeddings as a zero vector. It might not be very good for decoding since 'UNK' appears a lot in training examples and it also has a probability to be generated. To distinguish it with the pre-padded zero vectors, we set 'UNK' as tensorflow variable as well.

3. CNN

In both encoder and decoder, we have 6 convolution layers, each with filter width $k = 5$, $stride = 1$. The feedforward Neural network(with bias) is to ensure the dimension of location and scale of $q(z|x)$ to be the same as latent vector z. The hidden vector size is 50, and latent vector size = 32.

4. Optimization and Input

Optimizer: Adam

Learning rate: 0.001, with decay factor = 0.99 every 1000 steps

Number of Training Examples: 162419

Batch Size: 32

4 Results and Discussion

We report the Loss, negative log-likelihood and KL-divergence term for the training data set. The average loss (loss devided by batch size and number of iterations per epoch) decays from 90.532 to 73.562. Unfortunately, we also observe the 'KL' collapse phenomena: the KL penalty term decays fast to zero, while the negative log-likelihood term rarely decays after 10k steps.

We also tried adding random normal noise with std = 0.01 to the embedded word vectors. However, the result does not show benefit in doing that. Adding higher noise could slow

down the rate of KL divergence decay, but the total loss becomes higher as well.

We also tried generation of text. Here are some exampled sentences we generate using the sentences for movie reviews in HW3:

```
it is a is movie - is a rotten , rah - funny luhrmann , - actress.
director starring is actor - movie film with - - comedy, 4w 4w 4w 4w.
a enjoyable , humor film with really comedy a is - with 4w 4w 4w.
```

We notice that the generated text contains some information, but still doesn't look very much like English sentences. The possible reason is that the shifting trick is not enough, whereas (Yang et al., 2017) with more dilations and LSTM encoder have better performance. Also, (for time saving and parameter tuning) the 20k training sentences and 17615 size are not enough to learn a satisfactory model. We need more careful tuning of the structure of CNN layers as well as noise adding.

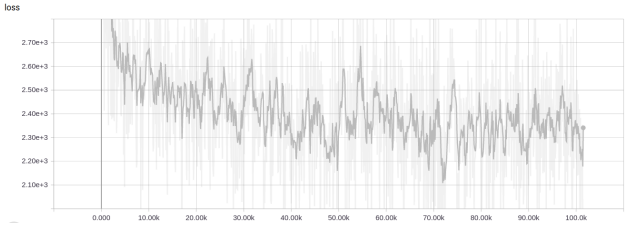

Figure 4: Total Loss

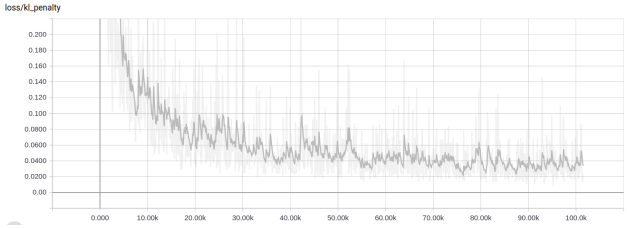

Figure 5: KL penalty term

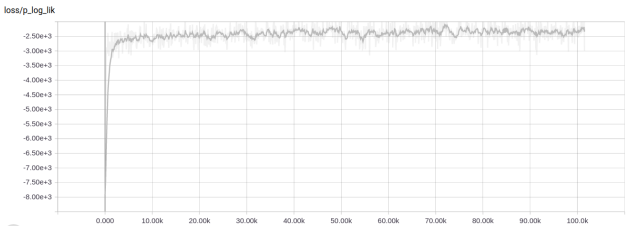

Figure 6: Log-likelihood

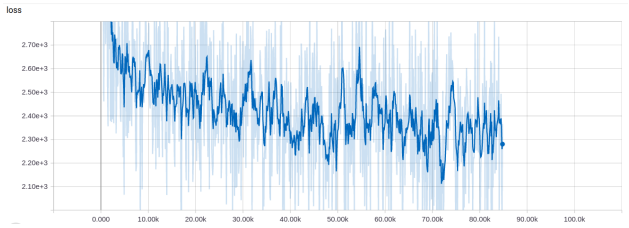

Figure 7: Total Loss (adding noise)

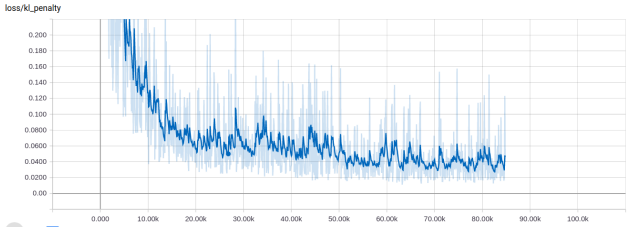

Figure 8: KL penalty term (adding noise)

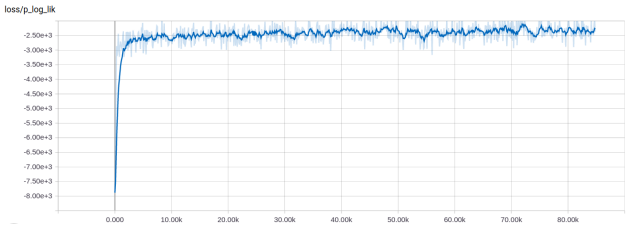

Figure 9: Log-likelihood (adding noise)

References

[Bowman et al.2015] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

[Doersch2016] Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

[Hu et al.2017] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

[Kingma and Welling2013] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Miao et al.2016] Yishu Miao, Lei Yu, and Phil Blun-

som. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

[Semeniuta et al.2017] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv:1702.02390*.

[Yang et al.2017] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *arXiv preprint arXiv:1702.08139*.