CSE 291E: Advanced Statistical NLP

Assignment 2: Deep Language Modeling

Due midnight on 05/25/2020

1 Introduction

In this assignment we will be carrying out an empirical investigation of how latent variables and recurrent neural networks can improve language modeling performance. We will mainly be working with a model class known as the variational autoencoder (VAE) [3, 1].

1.1 Recurrent Neural Network Language Model (RNN-LM)

As a baseline for this project, you will also be training a recurrent neural network (RNN) language model (RNN-LM). RNNs are a type of neural network that process inputs sequentially. Specifically, for an input sentence of n words, $x = (x_1, \ldots, x_n)$, an RNN will produce a sequence of n vector representations, $h_1, \ldots, h_n$, conditioned an an initial vector h_0, which is typically part of the RNN's parameterization. Each h_t is computed as a function of h_{t-1} and x_t. Since the same function is used at each time step, the network is called 'recurrent'. You will have a choice of using either a basic RNN, a variant called a GRU, or a variant called an LSTM – the latter two use a modified recurrence function that can improve performance. For more information about RNN variants please refer to:

`https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21`

`https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Further, a fun tutorial about all the cool things a RNNs can learn can be found here:

`http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

An RNN-LM uses the fact that each h_t is a function of words $x_1, \ldots, x_t$ and the RNN's parameters. As discussed in class, an RNN-LM treats h_t as a feature representation of the history up to word x_t, and then uses a logistic regression classifier based on this representation to form the conditional probability of the next word, $p(x_{t+1}|x_1, \ldots, x_t)$. An RNN-LM is trained by using gradient ascent to maximize the log likelihood (also called cross-entropy) of a training set of sentences with respect to the RNN parameters and the output logistic regression classifier parameters. The code we provide for this project will allow you to train an RNN-LM. Further, as discussed in the next section, the RNN-LM is a component of the VAE for text we will be working with.

1.2 Variational Autoencoder (VAE)

VAE refers to a combination of a deep latent variable model (shown in the right half of Figure 1) with a learning and inference technique known as amortized inference. The model introduces a continuous latent vector, z, which is generated from a Gaussian prior. An observation, x, is typically generated via a neural network, conditioned on z. VAE has been shown to be highly successful at vision tasks like image generation. VAE's application to language modeling has also been successful, but is generally more difficult. We will explore variants of the VAE model and training procedure and demonstrate under what conditions they yield improved language models.

For language modeling, an individual observation will be a single sentence, x. Thus, each sentence in the training corpus corresponds to its own latent z. However, all words in a sentence share the same latent z. VAEs for text typically use an RNN-LM to generate x conditioned on z. We will use this approach in this project. The code we provide lets you train an RNN-based VAE using amortized inference. Amortized inference introduces an inference network RNN that attempts approximate the model's true posterior, $p(z|x)$, using $q(z|x)$ in order to compute a lower bound on data marginal likelihood. Your first task will be to derive this lower bound as described in the next section.

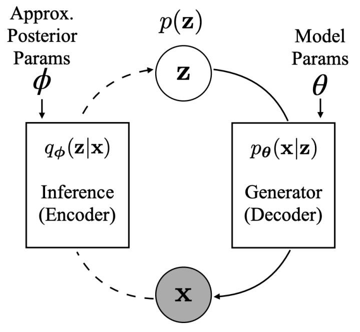

Figure 1: VAE [1]

2 Tasks

In this section we enumerate the steps to complete this project. The first part of your writeup should include a quick mathematical derivation as discussed next.

2.1 Derivation of ELBO

Since VAE introduces a latent vector z, the ideal training objective would be the log marginal likelihood of the training data, $\log p_\theta(x) = \log \int p_\theta(x|z)p(z)\mathrm{d}z$. When $p_\theta(x|z)$ is parameterized by a neural network like an RNN, this is typically intractable. Standard VAE training uses a lower bound on the marginal as an approximate objective. A tutorial on this approximation and how to train VAEs can be found here:

`https://jaan.io/what-is-variational-autoencoder-vae-tutorial/`

A popular approximate objective for training VAEs is known as the Evidence Lower Bound (ELBO) of $\log p_\theta(x)$. In this assignment we would like you to derive ELBO by showing the that following relationship holds for all distributions $q(z|x)$:

$$\log p_\theta(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \tag{1}$$

Here, $q_\phi(z|x)$ is often called an inference (or recognition) network and takes parameters ϕ. When the objective above is maximized, $q_\phi(z|x)$ forms an approximation to the model's true posterior, $p_\theta(z|x)$, which is intractable to compute. $p_\theta(x|z)$ represents the generator network with parameters θ, while D_{KL} stands for Kullback–Leibler divergence. You are required to include the derivation in the beginning of the writeup you submit for this assignment. As described in class and the tutorial above, VAE training is typically accomplished by using gradient ascent to optimize ELBO with respect to both θ, the model parameters, and ϕ, the inference network parameters. This is the approach we will use in this project.

2.2 Empirical Investigation

You are provided open source code for training an RNN-LM and an RNN-based VAE using PyTorch. However, you a free to implement both these models yourself or use any other open source codebase available. We will be using the Penn TreeBank (PTB) dataset for training and testing our models. The repository we suggest contains code for downloading PTB. See Section 4 below.

[1]Figure taken from [2]

The basic requirements for this project are that you (1) train an RNN-LM, (2) train an RNN-based VAE, and (3) that you compare results (e.g. validation or test likelihood) for the two models as you attempt to improve the VAE with some simple modifications. We expect you to include plots in your writeup as part of your investigation. For extra credit, you can dig deeper into VAEs to implement more advanced methods that help them train better.

The basic experiments that you will run and describe in your report (along with the ELBO derivation) are as follows:

1. Train an RNN-LM on the PTB dataset. You can use early-stopping to terminate the training. Show your training and validation/test loss plots. Your loss plots correspond to negative log likelihood, and thus can tell you how well your model generalizes to held out data.
2. Train an RNN-based VAE on the same dataset and show your training and validation/test loss plots. Your loss plots correspond to negative ELBO. Since ELBO is a lower bound on log marginal data likelihood, it can be used as an approximation to your model's held out performance.
3. Did your VAE training result in posterior collapse (see lecture slides for a description of this issue)? Provide empirical evidence to justify your observation. Does your VAE outperform the RNN-LM? Can you tell from ELBO alone?
4. Propose, justify, and implement one modification to your VAE implementation that may improve held-out likelihood (e.g. introduce an annealing schedule for the KL term as described in class, OR change your encoder or decoder architecture, OR change your gradient update schedule, etc.). Did your modification work or not? Provide empirical evidence.

2.3 Extra Credit

You may choose to carry out any or none of the following extra credit options:

1. Researchers have proposed many ways to avoid posterior collapse. Two generally successful methods are explained in [2] and [4]. Implement one or both of these approaches. Compare the performance of the model before and after this change. You can use publicly available code as long as you reference the sources in your report.
2. Ideally, the latent variable, z, provides the decoder with high level information about what the final generated output will contain. For example, if you were training a VAE to do image generation for indoor scenes, we might hope for z to capture the lighting condition of the scene overall (e.g. bright overhead light, multiple point sources, etc.). Similarly, on text, we might hope that z captures something like the overall style of the sentence being generated. Unfortunately, current VAE models are difficult to control and it is often hard to predict what z will capture.

   For this task, you will create a new dataset (mix-dataset) that is a mix of sentences from datasets of different domains or styles (dataset1, dataset2) – e.g. tweets and financial newspaper content, or yelp reviews and amazon reviews, etc. Train a VAE on this mixed dataset and then visualize the latent space in two dimensions using t-SNE or PCA. If you color the embedded sentences by which original dataset they came from, do you see any evidence of separation in the latent space? Propose an implement other analyses for discovering what z captures.
3. Choose your own adventure! Propose and implement your own analysis or extension of deep language models.

3 Implementation Tips

We suggest forking the following github repository, which has code for a VAE and RNN Language Model, along with setup/execution instructions: `https://github.com/hammad001/Language-Modelling-CSE291-AS2`. However, you are free to use any open source codebase as long as you cite it in your report, or implement everything yourself from scratch in the framework of your choosing.

An alternative implementation with more advanced features can be forked from: `https://github.com/jxhe/vae-lagging-encoder`. You are not required to turn in your final code – only turn in your writeup.

References

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

[2] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *CoRR*, abs/1901.05534, 2019.

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

[4] Jiacheng Xu and Greg Durrett. Spherical latent spaces for stable variational autoencoders. *CoRR*, abs/1808.10805, 2018.