Particle-filter based language models

November 20, 2017

Rachit Singh, Yuntian Deng
rachitsingh@college.harvard.edu

We were discussing methods for inducing multimodality into tasks like translation or summarization. A lot of approaches are looking at forming a distribution over latent vectors that encode the entire sentence together, but we decided to look at the problem from the perspective of localized changes. So, instead of taking a variational distribution $q(z)$ over the hidden state before passing through a decoder, we thought it might be interesting to instead draw at each time step from a distribution $z_t \sim q_t(z_t; \lambda_t)$ and form the generated word from the distribution $p(y_t|z, y_{1:t-1})$.

This has the downside of not leading directly to a "sentence representation", since the sequence of encoded vectors is as long as the sentence itself, but our guess is that generation is more natural, if we can find the right representation q. For example, if we initialize the prior distribution of $\mathbf{z}$ to be the empirical distribution of part-of-speech tag sequences we find in the training data, then we might softly contrain the model $\mathbf{q}$ to be close to these sequences.

Generative Model

To formalize the idea, we have the following generative process:

$$\mathbf{z} \sim p(\mathbf{z}) = \prod_{t=1}^{m} p(z_t \mid \mathbf{z}_{1:t-1}) \tag{1}$$

$$\mathbf{y} \sim p(\mathbf{y}|\mathbf{z}) = \prod_{t=1}^{m} p(y_t \mid \mathbf{y}_{1:t-1}, z_t) \tag{2}$$

$$z_t \sim p(z_t|\mathbf{z}_{1:t-1}) = \mathrm{Cat}(\lambda_t^z) \tag{3}$$

$$y_t \sim p(y_t|z_t, \mathbf{y}_{1:t-1}) = \mathrm{Cat}(\lambda_t^y) \tag{4}$$

Where the parameters λ_t^z, λ_t^y are computed using recurrent neural networks. Assuming that $\mathbf{h}_{\rightarrow}, \mathbf{h}_{\leftarrow}$ are the hidden states corresponding to the traversal of a bidirectional RNN over $\mathbf{y}$, we can compute a sequence of hidden states h_t^e, h_t^d:

$$\begin{aligned}
h_t^e &= \mathrm{RNN}(h_{t-1}^e, h_{\rightarrow,t}, h_{\leftarrow,t}, z_{t-1}) \\
\lambda_t^z &= f(h_t^e) \\
h_t^d &= \mathrm{RNN}(h_{t-1}^d, h_{\rightarrow,t}, h_{\leftarrow,t}, y_{t-1}) \\
\lambda_t^y &= g(h_t^d)
\end{aligned}$$

where RNN corresponds to any RNN structure (e.g. a GRU or LSTM), and f, g might correspond to a linear map into the output space. The BiRNN could be replaced with a regular RNN to reduce complexity. This is not equivalent to a state space model as considered by (**?**), since for that model the transition distribution for z_t only depends on the previous z_{t-1}, and the emission distribution for y_t only depends on the latent variable z_t. We think that such a simple model doesn't capture the dependencies of natural language.

Inference

We can use several different variational approximations that might lead to more powerful approximation. The straightforward approach is to use

$$q(\mathbf{z}|y) = q(z_1|y) \prod_{i=2}^{L} q(z_i|z_{i-1}, y)$$

One potential change is that we can factor the variational distribution using what is called a 'top-down variational approximation': we can allow

$$q(\mathbf{z}|y) = q(z_L|y) \prod_{i=1}^{L-1} q(z_i|z_{i+1}, y)$$

(note that the posterior is factored in reverse). **?** uses this technique in *ladder variational autoencoders* (LVAE). **?** also indicates that doing inference in reverse might make more sense.

While we have a somewhat clearer sense of what the generative model should look like, we have a lot of choice over *what* our hidden variables are. For example, we can assume that they they're Gaussian, or possible a discrete random variable.

We also have significant choice over the prior in these case. Since we want the prior to be over the entire vector $\mathbf{z}$, we can factor the prior over time as well.

Particle Filter

Recent work (**??**) has shown how to find a significantly tigher variational bound in the case of sequential latent distributions (see fig. 1). Essentially, by treating the sequence of samples (i.e. collection of samples for z_1, another as z_2, etc.) as the progressive states of a particle filter, we can do better than the naive approach of sampling k separate chains, by using a *resample/reweight* step to remove low weight sequences and resample high weight sequences. In effect, this is a form of probabilistic, differentiable beam search - we can essentially optimize the proposal distribution $p(z_t|\mathbf{z}_{1:t-1}, \mathbf{y})$, but only if the corresponding predictions are good.

There's also some room for exploration here, since both implementations above appear to use black-box variational inference for the *resample/reweight* step, since the choice of which particles to use as an ancestor is discrete, and it's not acceptable to have 'points in the simplex' as in the Concrete distribution. However, the paper we recently read in reading group (Backpropagation through the Void) might prove better suited for this task, which is something to explore given time.

Disadvantages

One primary disadvantage is that the addition of another sequential model (i.e. a decoder) for predicting the latent state distribution increases the number of parameters significantly. Our approach for dealing with this concern is to reduce the size of the latent representation $\mathbf{z}$. Since we're stretching the latent representation over the entire sequence, we can let $z_t \in [k]$ rather than $z_t \in \mathbb{R}^k$, and reduce the amount of inference necessary. In general using a discrete latent space is difficult for inference, but recent advancements have made this problem less difficult. Still, since we can only sample a very small subset of the possible sequences (and they are totally disjoint), we might have trouble early in the training process.

Advantages

1. **A tighter approximation to the marginal might lead to multimodality in our responses**: The reason is this: suppose that we're in the autoencoder setting, and we have derived parameters $\lambda = f(\mathbf{x})$ for the variational distribution $q(\mathbf{z}; \lambda)$. If we take multiple samples $\mathbf{z}_1, \mathbf{z}_2, \dots \mathbf{z}_k$, and calculate the log-joint $\log p(\mathbf{x}, \mathbf{z}_i)$ for each of them, we'll get several values. If we're properly multimodal, then one of the $\mathbf{z}_i$ might correspond to a exact phrasing of $\mathbf{x}$ that we have (rather than a rearrangement or rephrasing) and so $\log p(\mathbf{x}, \mathbf{z}_i)$ might be high, while all of the other log-joints are very small. Then, taking the log-sum-exp of these values (corresponding to $\log \sum_i^k p(\mathbf{x}, \mathbf{z}_i)$) will still be high, even though most of the other predictions are bad, which is what we want. So our usual metric of the ELBO would score this poorly, even though this metric (the IWAE bound) would score it high. In particular, as $k \to \infty$, the evaluation gets closer to what we want, and also the bound goes to the true marginal $\log p(x)$.

2. **It might be more interpretable**: in general, it's difficult to interpret the latent space learned by a single vector $\mathbf{z}$. Even though the latent sequence $[z_1, z_2, \dots, z_m]$ is not more likely to be interpretable, it's possible that we can force it to be so. For example, we might constrain training on this portion of the model early by letting $\mathbf{z}$ be exactly the POS tags for the sequence $\mathbf{y}$, and then train the inference network later once we have 'locked in' what the indicator z_t mean. Alternatively, we can restrict the generative model $p(y_t \mid z_t \dots)$ to only generate words that correspond to the POS tag z_t (again, from the true $\mathbf{y}$) to bootstrap training. Another idea is to constrain $p(\mathbf{z} \mid \mathbf{x})$ to be limited to sequences that we've seen in the training data. This also tackles the perpetual issue of identifiability of neural models.

Related work

Unfortunately this section is a little less fleshed out than it should be. Much of the work on using variational RNNs focuses on state space models (**????**). The main analogue of this work is the neural variational translation model of **?**), which uses a single latent state for translation, and **?**, which also uses sequential latent variables for the task of summarization, but which use Gaussian latent variables and don't tackle the identifiability concern as above, and don't use the particle-filter bound to tighten the objective.

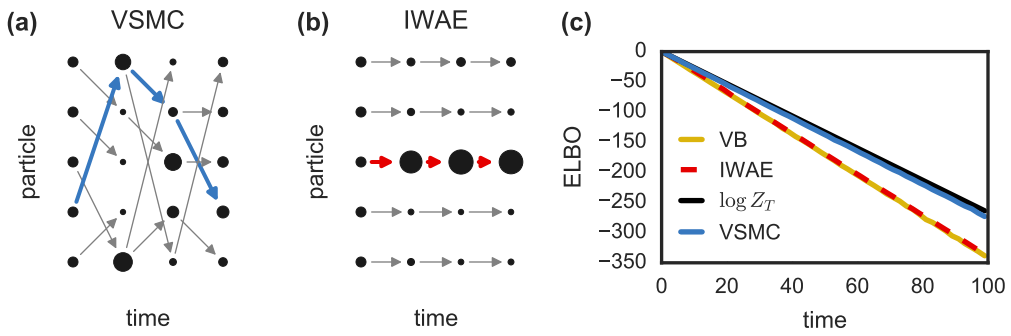

Figure 1: Comparing VSMC and the IWAE (from (**?**)) (a) VSMC resamples the particles at each step, and chooses the most likely sequence, akin to beam search. (b) IWAE does the same, but without resampling, and only chooses one of the k chains. This means that only one sequence might be sampled in general. (c) As time increases, IWAE and regular variational Bayes (VB) diverge from the true marginal likelihood, while VSMC stays close.

[side note: **?** say they are beating state of the art on summarization, but isn't the pointer-generator networks SOTA right now? they don't compare]

Plan of attack

Let me know if this makes sense (my intuition is not great):

1. Build an autoencoder that uses the above generative model and inference network, and train it to work without a tighter bound on PTB [1-2 weeks]
2. Add attention if that makes sense [1 week?]
3. Add the particle filter bound, and measure the improvement [2 weeks]
4. Spend a lot of time trying variants [winter break]