Tutorial 10 (20 pt.) Combining Variational Inference and Sequential MC methods using python

Submit Assignment

Due No Due Date **Points** 20 **Submitting** a file upload
File Types pdf, txt, zip, gzip, ipynb, and py
Available Jan 14 at 12am - Mar 13 at 11:59pm about 2 months

Excellent tutorial but theoretically advanced and state of the art. Tutorial 11 is should be done before this. Unless you know more than me you will need to read a lot of papers and think hard about them for some time before you understand this. For me it was worth it.

Maybe start by understanding this:

IWAE.pdf

Perhaps you might need to brush up on MCMC:

Andrieu_et_al-2010-Journal_of_the_Royal_Statistical_Society%3A_Series_B_%28Statistical_Methodology%29 (1).pdf

This is the paper that you must understand more or less completely to do the tutorial:

[1] Filtering Variational Objectives, Maddison et al.

7235-filtering-variational-objectives.pdf

I found that for the practical implementation details I needed to read this too:

VRNN Paper: A Recurrent Latent Variable Model for Sequential Data Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, Yoshua Bengio

VRNN.pdf

So you should, as I did, be proactive in searching for whatever you need to read to understand this. I actually read more than this but I think this is what is needed.

Here are the tutorial files:

vismc_pgm_tutorial_theory.pdf

misc_vismc_tutorial.ipynb.pdf

fivo (1).zip

For the review explain approximately how the distributions are being modeled (and what they are modelling).

For example connection of the graphs in Fig.1 of the VRNN paper to the computations of algortihm 1 in [1]. That will take some time so do not start there and try to work back but rather try to understand generally with that specific example as the final ah ha moment.One should do Olga's VI tutorial first.

One needs to read the paper [1] Filtering Variational Objectives too. I needed to read other references as well. Google terms you do not understand.

Let me try to give some insights at the risk of being wrong myself about some of this.

The key trick to all of this is that one can move the gradient past the expectation using the reparametrization trick as described in the IWAE paper. or look here:

http://kvfrans.com/variational-autoencoders-explained/

Some of the refs come at this from a variational inference perspective and use that jargon. Other are more NN people and tend to talk using an implied NN always present.

Variational autoencoders, VAE, the 'encoder model' is also referred to as the recognition model. It generates a distribution over the latent variables from observed data, typically the mean and standard deviation of a Guassian is generated by a NN. The 'decoder model' is part of the 'generative model' and takes a sample from the latent variable distribution and produces a sample that should look like the data.

A confusing thing is the 'names' of these different approaches. Notice some paper's refer to an IS, Importance Sampling' method, but importance sampling is actually part of all these methods, IMO. There is sampling at several levels here. One is to compute the expectation. That one is sort of hidden but each iteration represents one such sample of all latent states 1 to T. So in [1] eq (4) is going to be actually evaluated by summing samples (ie Monte Carlo expection) where each sample is a whole set of particles each from 1 to T drawn from q. All with the same observed x. Then we take another x and repeat In practice we want to max it so we do a tiny gradient step between moving to the next x. As in back propagation. (At least that is what I think is being done although one could do batches.)

Then there are samples along the way and resampling of the particles which are only part of the latent state up to time t. Extra confusion when we get to the implementation is the introduction in RNN of a 'hidden' state which is sort of an agumentation of the latent state (which is also hidden). This hidden state encodes all the past values of the latent and observed varibles so that they do not appear explicitly in the model but rather it ends up looking like a first order markov process. All the past dependence must be in these hidden states.

Remember that even if we generate distributions over 'x' we always observe it and do not sample it. These distributions are needed so we can use the value of the probability density at that x_t (and a sampled z^i_t (and hidden state)) in computing the importance weight, alpha.

I was perplexed as to the why Algorithm 1 in [1] counts the unnormalized weights for particles that later are lost during resampling. These should be replaced by additional copies of the duplicated samples according to the path traced back by each particle. But no the formula is correct and one does use the particles that die in this way to get an unbiased estimator for p(x).

This is explained very well in Equation (9) of The classic paper: Particle Markov chain Monte Carlo methods, Andrieu et al. (9) is simple enough and is just the definition of a joint, factored as conditionals. Then below when you substitute in the expression for each conditioal from the unnormalized weights you see why Algorithm 1 of [1] works.

The other big mystery here is the extra gradient terms for the resampling step. All papers [1-3] have it and all say it gives high variance and do not use it. So I believe it. I just do not know how they reason to get the expression for it but it seems to be consistent accross many sources.

On running the code:

Notice that IWAE is just FIVO with no resampling and 'ELBO' is IWAE with only one sample (Sample here means one for the full hidden state vector so lots of sampling along the way. It seems that what is called 'ELBO' here is is just a way of optimizing the ELBO using stocastic Gradient ascent. That is take a sample, compute the gradient, and move uphill.

Note that when running the training 'checkpoints' are stored in /tmp/fivo-jsb

Those are binary files that tensor flow understands.

If you want to train again for example with a different bound you need to delete those files or move them. Otherwise it seems to start from the last checkpoint.

For ex 5 it is easiest to not try to modify the def for the 3 criteria to accept the new needed arguments and then create a new def for the new criteria. Rather do the ugly: go into line 2xx and hard code what you want there as several lines of code that change how that section works. Use Rika's hints on the tensorflow function to use.