Supplement to: Inferring Graphics Programs from Images

Anonymous Author(s)
Affiliation
Address
email

1 Neural networks for guiding SMC

Let $L(\cdot|\cdot)$: image$^2 \to \mathcal{R}$ be our likelihood function: it takes two images, an observed target image and a hypothesized program output, and gives the likelihood of the observed image conditioned on the program output. We want to sample from:

$$[p|x] \propto L(x|\text{render}(p))[p] \tag{1}$$

where $[p]$ is the prior probability of program p, and x is the observed image.

Let p be a program with L lines, which we will write as $p = (p_1, p_2, \cdots, p_L)$. Assume the prior factors into:

$$[p] \propto \prod_{l \leq L} [p_l] \tag{2}$$

Define the distribution $q_L(\cdot)$, which happens to be proportional to the above posterior:

$$q_L(p_1, p_2, \cdots, p_{L-1}, p_L) \propto q_{L-1}(p_1, p_2, \cdots, p_{L-1}) \times \frac{L(x|\text{render}(p_1, p_2, \cdots, p_{L-1}, p_L))}{L(x|\text{render}(p_1, p_2, \cdots, p_{L-1}))} \times [p_L] \tag{3}$$

Now suppose we have some samples from $q_{L-1}(\cdot)$, and that we then sample a p_L from a distribution proportional to $\frac{L(x|\text{render}(p_1,p_2,\cdots,p_{L-1},p_L))}{L(x|\text{render}(p_1,p_2,\cdots,p_{L-1}))} \times [p_L]$. The resulting programs p are distributed according to q_L, and so are also distributed according to $[p|x]$.

How do we sample p_L from a distribution proportional to $\frac{L(x|\text{render}(p_1,p_2,\cdots,p_{L-1},p_L))}{L(x|\text{render}(p_1,p_2,\cdots,p_{L-1}))} \times [p_L]$? We have a neural network that takes as input the target image x and the program so far, and produces a distribution over next lines of code (p_L). We write $\text{NN}(p_L|p_1, \cdots, p_{L-1}; x)$ for the distribution output by the neural network. So we can sample from NN and then weight the samples by:

$$w(p_L) = \frac{[p_L]}{\text{NN}(p_L|p_1, \cdots, p_{L-1}; x)} \times \frac{L(x|\text{render}(p_1, p_2, \cdots, p_{L-1}, p_L))}{L(x|\text{render}(p_1, p_2, \cdots, p_{L-1}))} \tag{4}$$

Then we can resample from these now weighted samples to get a new population of particles (here programs are particles), where each program now has L lines instead of $L - 1$.

This procedure can be seen as a particle filter, where each successive latent variable is another line of code, and the emission probabilities are successive ratios of likelihoods under $L(\cdot|\cdot)$.

Comments for Dan. Right now I'm not actually sampling from the neural network - instead, I enumerate the top few hundred lines of code suggested by the network, and then weight them by their likelihoods. So actually the form of NN is:

$$\text{NN}(p_L|p_1, \cdots, p_{L-1}; x) \propto \begin{cases} 1, & \text{if } p_L \in \text{top hundred neural network proposals} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2017). Do not distribute.

Algorithm 1 Neurally guided SMC

Input: Neural network NN, beam size N, maximum length L, target image x
Output: Samples of the program trace
Set $B_0 = \{\text{empty program}\}$
for $1 \leq l \leq L$ **do**
  for $1 \leq n \leq N$ **do**
    $p_n \sim \text{Uniform}(B_{l-1})$
    $p'_n \sim \text{NN}(\text{render}(p), x)$
    Define $r_n = p'_n \cdot p_n$
    Set $\tilde{w}(r_n) = \frac{L(x|r_n)}{L(x|p_n)} \times \frac{[p'_n]}{[p'_n = \text{NN}(\text{render}(p), x)]}$
  end for
  Define $w(p) = \frac{\tilde{w}(p)}{\sum_{p'} \tilde{w}(p')}$
  Set B_l to be N samples from r_n distributed according to $w(\cdot)$
end for
return $\{p : p \in B_{l \leq L}, p \text{ is finished}\}$

Do you think this is a problem? The neural network puts almost all of its mass on a few guesses. In order to get the correct line of code I sometimes need to get something like the 50th top guess, so I don't want to literally just sample from the distribution suggested by the neural network.