GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution

Matt Kusner, Jose Miguel Hernandez-Lobato

Satya Krishna Gorti

University of Toronto

Introduction

- In the GAN methodology, we have a Generator network G, and a Discriminator network D.
- The Discriminator is used to predict whether a data instance is synthetic or real. The Generator G is trained to confuse G by training high quality data.
- GANs are trained by propagating gradients backward from D to G.
- This is only feasible if generated data is continuous.

- Discrete data, encoded as one-hot representation, can be sampled from a categorical distribution, however this sampling process is not differentiable.
- We can however obtain a differentiable approximation from the Gumbel-Softmax distribution.

The Gumbel distribution trick

- The CDF of a standard Gumbel distribution is given by $CDF_{gumbel}(g_i) = exp(-exp(-g_i))$
- A random variable g_i is said to have a Gumbel distribution if $g_i = -\log(-\log(U))$ where $U \sim Uniform[0,1]$ (Inverse transform sampling)
- The Gumbel-Max trick is a method to sample from a categorical distribution $Cat(\alpha_1, \alpha_2, ...\alpha_K)$

- let set $\{g_k\}_{k \leq K}$ be an i.i.d sequence of standard Gumbel random variables.
- The trick is based on the observation that $k_{max} = \arg\max_k(\log \alpha_k + g_k)$ follows the desired categorical distribution. (Proof at blog post by Ryan Adams [1])
- Hence a discrete variable sampled from the categorical distribution can be written as:
  $y = onehot(\arg\max_k(\log \alpha_k + g_k))$
- Therefore, procedure to sample from a categorical distribution is:
  - draw Gumbel noise by transforming uniform random samples.
  - add it to $\log \alpha_k$ (α_k can be unnormalized probability)
  - take value of k that produces the maximum.

[1] https://hips.seas.harvard.edu/blog/2013/04/06/the-gumbel-max-trick-for-discrete-distributions/

Gumbel-Softmax relaxation trick

- Since, $\arg\max$ operator is not continuous, we need a differentiable approximation.
- The Gumbel-softmax trick is to approximate the operator with a softmax transformation.
- We approximate y with:
  $y_k = \frac{exp((\log \alpha_k + g_k)/\tau)}{\sum_{i=1}^{K} exp((\log \alpha_i + g_i)/\tau)}$

Example problem

- The GAN is made to approximate sequence given by the below CFG. $S \rightarrow x \mid S + S \mid S - S \mid S * S \mid S/S$, where x is the terminal.
- Examples of valid strings:
  $x * x + x - x/x * x + x + x$,
  $x + x$,
  x

RNN for discrete sequences

- The generative model is based on an LSTM RNN

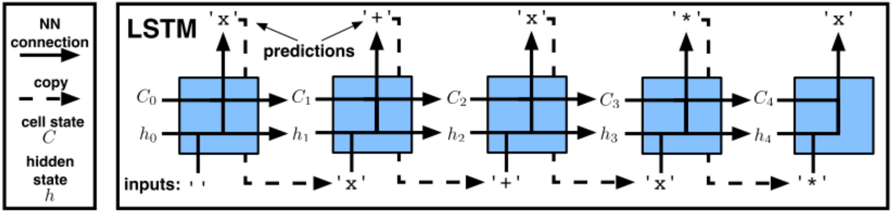


Figure: A classic LSTM RNN model during the prediction phase

- The LSTM RNN is trained to predict a hidden-state vector **h** at every time step, the *softmax* operator is then applied to h which gives us a distribution over all possible generated characters (in our case: $x, +, -, /, *$)
- LSTM model is trained by matching the softmax distribution to a one-hot encoding of the input data via Maximum Likelihood Estimation (MLE).
- We need to build a generative model for discrete sequences, which is accomplished by sampling through the LSTM using GAN approach

- In this case, both Generator G, and Discriminator D are LSTMs with parameters Θ and Φ respectively.
- The Generator network G takes as input a sample-pair which effectively replaces the initial of the hidden and memory cell states.

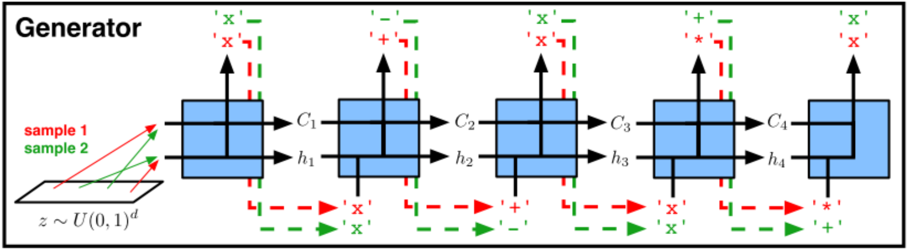


Figure: Generator network for discrete sequences

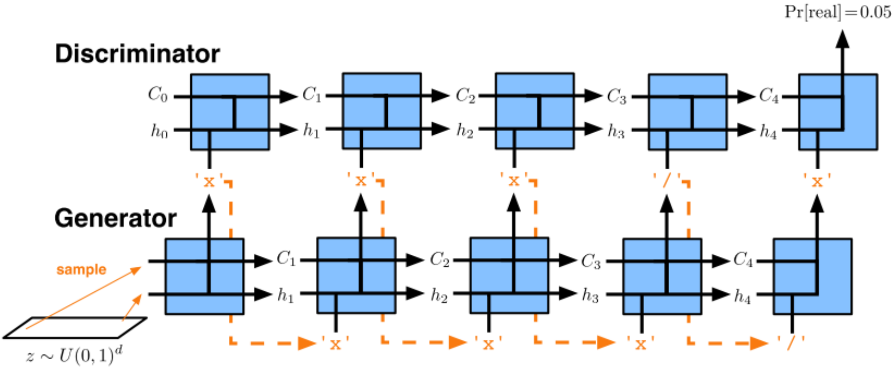


Figure: The GAN network

- Our aim while training the GAN is to minimize differentiable loss functions for G and D to update Θ and Φ.
- Since we already know that sampling points from G from the categorical distribution given by the LSTM is not differentiable.
- We can now use the Gumbel-softmax distribution to sample from LSTM and optimize Θ and Φ using back-propagation.
- i.e. Instead of

$$y = onehot(\arg\max_k(h_i + g_i)) \tag{1}$$

we instead use,

$$y = softmax((h + g)(1/\tau)) \tag{2}$$

when $\tau \to 0$, we have same distribution of that generated by (1)
when $\tau \to \infty$, the samples are always from uniform probability vector.

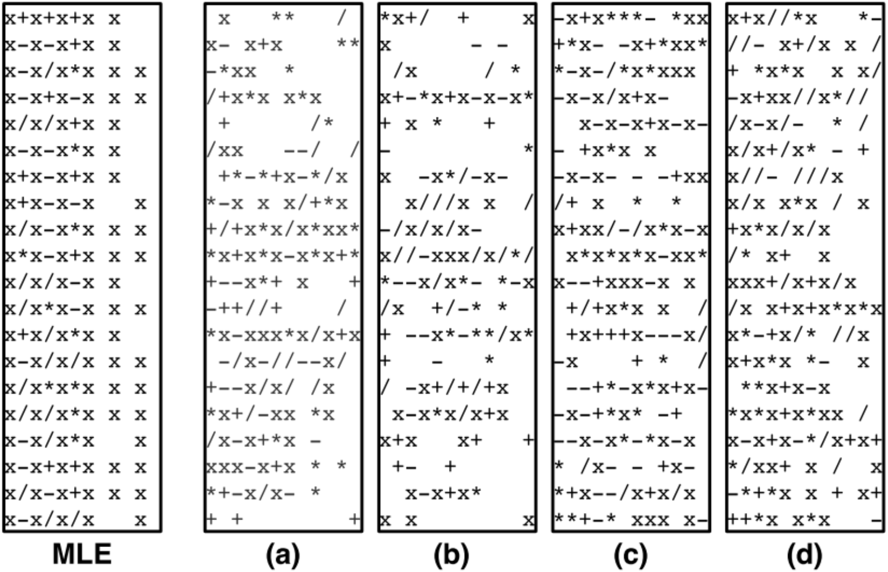
Algorithm

data: $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \sim p(\mathbf{x})$,
Generative LSTM network G_Θ
Discriminative LSTM network D_Φ
while loop until convergence **do**
    Sample mini-batch of inputs $B = \{\mathbf{x}_{B_1}, \ldots, \mathbf{x}_{B_m}\}$
    Sample noise $N = \{\mathbf{z}_{N_1}, \ldots, \mathbf{z}_{N_m}\}$
    Update discriminator $\Phi = \text{argmin}_\Phi -\frac{1}{m}\sum_{\mathbf{x}\in B} \log D_\Phi(\mathbf{x}) - \frac{1}{m}\sum_{\mathbf{z}\in N} \log(1 - D_\Phi(G_\Theta(\mathbf{z})))$
    Update generator $\Theta = \text{argmin}_\Theta -\frac{1}{m}\sum_{\mathbf{z}\in N} \log \frac{D_\Phi(G_\Theta(\mathbf{z}))}{1 - D_\Phi(G_\Theta(\mathbf{z}))}$
end while

Experimentation

- 5000 samples with maximum length of 12 characters were generated from CFG defined previously. (All sequences with less than 12 characters were padded with spaces)
- Trained G and D for 20,000 mini-batch iterations.
- Linearly annealed temperature coeff from $\tau = 5$ to $\tau = 1$.

Results



Thank you