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

Authors: Matt Kusner, Jose Miguel Hernandez-Lobato

Presented by Shehzeen Hussain
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 $D$ 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**.

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)} \left[ \log D(x) \right] + \mathbb{E}_{z \sim p_z(z)} \left[ \log (1 - D(G(z))) \right]$$
Task: Can we train a GAN to generate Discrete Sequences

- Given a dataset of Real-world sequences, can we train a Generative Adversarial Network to learn to generate sequences that are indistinguishable from the real sequences?
GANs for Discrete Sequences

This is a Real Sequence

Real/Fake?

Discriminator

T
H
E

t_1
t_2
t_3

Generator

z
GANs for Discrete Sequences

Is there a Problem somewhere while training such a setup?

*Hint: Look down here*

This is a Real Sequence

Real/Fake?

Generator

Discriminator

\[ t_1 \quad t_2 \quad t_3 \]
GANs for Discrete Sequences

Is there a Problem somewhere while training such a setup?

Hint: Look down here

Sampling: Non-differentiable

This is a Real Sequence
A Closer Look at the problem

Softmax Distribution

Arg-max/Sampling Operation

Embedding Matrix - Pick an embedding vector
Gumbel-Softmax Trick

- Don’t sample, pass the whole distribution to the discriminator instead.
- Not the softmax distribution but a “Gumbel-Softmax Distribution”

\[
p = \text{softmax}(h) \quad [\text{softmax}(h)]_i = \frac{\exp(h_i)}{\sum_{j=1}^{K} \exp(h_j)}, \quad \text{for} \quad i = 1, \ldots, d.
\]

Sampling \(y\) from \(p\), is the same as doing the following:

\[
y = \text{one_hot}(\arg\max_i (h_i + g_i))
\]

Approximate \(y\) with a softer/continuous version:

\[
y = \text{softmax}(1/\tau(h + g)))
\]

\(g_i\) are independent r.v. and follow a gumbel distribution with 0 mean and unit scale

Temperature parameter controls the “flatness” of the distribution
Gumbel-Softmax Trick - Contd.

Temperature Parameter Effect:

Controls Flatness or Stochasticity of the Distribution

\[ y = \text{softmax}(1/\tau(h + g)) \]
Gumbel Softmax Trick - GAN

This is a Real Example

Real/Fake?

Discriminator

g
p
t_1 t_2 t_3

Generator

z
Gumbel Softmax Trick - GAN

Why can’t we pass $p$ directly into the discriminator? Why do we need gumbel-softmax trick?

This is a Real Example

Generator

Discriminator

Real/Fake?

$g$

$p$

$t_1$

$t_2$

$t_3$

$z$
g is parameterized by temperature and we anneal it during training.

This is a Real Example
Task - Synthetic Data language modeling

The GAN is made to approximate sequence given by the below CFG.
\[ S \rightarrow x \mid S + S \mid S - S \mid S \times S \mid S/S, \] where \( x \) is the terminal.

Examples of valid strings:
\[ x \times x + x - x/x \times x + x + x, \]
\[ x + x, \]
\[ x \]
Generator-Model (LSTM) in the generation phase

Figure: A classic LSTM RNN model during the prediction phase
Training Objective

- Traditionally an LSTM is trained using the Maximum Likelihood Estimate (MLE) objective to predict the next character given the previous characters so we use n-1 tokens to predict nth token. This suffers from Exposure-Bias problem.

- Now, we want to build a generative model for discrete sequences, which is accomplished using GAN approach
The generator model
The GAN network

Figure: The GAN network
Algorithm

1: **data:** \( \{x_1, \ldots, x_n\} \sim p(x) \),
2: Generative LSTM network \( G_\Theta \)
3: Discriminative LSTM network \( D_\Phi \)
4: while loop until convergence do
5: Sample mini-batch of inputs \( B = \{x_{B1}, \ldots, x_{Bm}\} \)
6: Sample noise \( N = \{z_{N1}, \ldots, z_{Nm}\} \)
7: Update discriminator \( \Phi = \arg\min_\Phi -\frac{1}{m} \sum_{x \in B} \log D_\Phi(x) - \frac{1}{m} \sum_{z \in N} \log (1 - D_\Phi(G_\Theta(z))) \)
8: Update generator \( \Theta = \arg\min_\Theta -\frac{1}{m} \sum_{z \in N} \log \frac{D_\Phi(G_\Theta(z))}{1 - D_\Phi(G_\Theta(z))} \)
9: 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 parameter from $\tau = 5$ to $\tau = 1$. 
Experiments

Figure 3: The generative and discriminative losses throughout training. Ideally the loss of the discriminator should increase while the generator should decrease as the generator becomes better at mimicking the real data. (a) The default network with Gumbel-softmax temperature annealing. (b) The same setting as (a) but increasing the size of the generated samples to 1,000. (c) Only varying the input vector temperature. (d) Only introducing random noise into the hidden state and not the cell state.
Results: Only qualitative

- Each row is a sample from either model, each consisting of 12 characters (included blank space character as some training inputs are padded with spaces if less than 12 characters).
- GAN models are learn to generate alternating sequences of x’s, similar to the MLE result. Specifically, the 4th, 10th, and 17th rows of plot (a), show samples that are very close to the training data.
- Included for reference: MLE LSTM is not strictly a generative model in the sense of drawing a discrete sequence from a distribution.
Concluding Remarks

- Theoretically sound, and Gumbel-softmax trick is straightforward to implement compared to other techniques such as REINFORCE and not many hyper-parameters to tune.
- Counters non-differentiability issue when using GANs for generating discrete sequences
- No convincing experiments on real data
- Lacking evaluation of using probability distribution $p$ instead of gumbel-softmax distribution $g$.

Motivates future research that can use Gumbel-softmax trick to address the non-differentiability issue faced with discrete data and using GANs for discrete data.
Related Works

- Neural Machine Translation with Gumbel-Greedy Decoding (AAAI 2018)
- Gumbel-Tree LSTM: Learning to Compose Task-Specific Tree Structures (AAAI 2018)
- Categorical Reparameterization with Gumbel-Softmax (ICLR 2017)
- Recent research using GANs for generating natural language expressions which capture diversity and richness inside sentences: RankGAN: Adversarial Ranking for Language Generation (NIPS 2017)
Thank you!
Backup Slides
Softmax Equation

- Terminology:
  - $z$ -> the set of input values
  - $z_j$ -> the jth element in the set of input values
  - $k$ -> the total number of input values

- Below is the equation for calculating the softmax value:

$$f(z_j) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

- Example:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$z$ = {8, 4, 2}</td>
<td></td>
</tr>
<tr>
<td>$\sum_k e^{z_k}$ = 2981 + 54.6 + 7.4 = 3043</td>
<td></td>
</tr>
<tr>
<td>$f(8)$ = 2981 / 3043 = 0.98</td>
<td></td>
</tr>
<tr>
<td>$f(4)$ = 54.6 / 3043 = 0.018</td>
<td></td>
</tr>
<tr>
<td>$f(2)$ = 7.4 / 3043 = 0.002</td>
<td></td>
</tr>
</tbody>
</table>

All values add up to 1