Detailed steps for training a neural editor

Kelvin Guu, Tatsunori Hashimoto, Yonatan Oren, Percy Liang

August 19, 2018

1 Introduction

- This document accompanies "Generating Sentences by Editing Prototypes".
- It provides more detailed instructions for training a neural editor, and uses all the same notation
- Implementation available on GitHub at: https://github.com/kelvinguu/neural-editor
- Reproducible experiments available on CodaLab at: https://bit.ly/2rHsWAX
- Spot an error in this document? Please let us know at kguu@google.com

2 Training objective

- Let $\Theta = (\Theta_p, \Theta_q)$ be the full set of parameters, where:
  - Θ_p is the set of parameters for the neural editor, $p_{\text{edit}}(x \mid x', z)$. This includes:
    * The parameters of the sequence-to-sequence encoder and decoder
    * A set of input word vectors (used by the encoder)
    * A set of output word vectors (used by the decoder in its softmax layer)
    * (Optionally, the input and output word vectors can be tied)
  - Θ_q is the set of parameters for the inverse neural editor, $q(z \mid x, x')$
    * This is just a set of word vectors, as described in Section 3.4 of "Generating Sentences by Editing Prototypes"
    * (Optionally, these word vectors can be tied with the input/output word vectors of the editor)
- The overall training objective is:

$$\begin{aligned} \mathcal{O}(\Theta) &= \sum_{x \in \mathcal{X}} \sum_{x' \in \mathcal{N}(x)} \text{ELBO}(x, x') \\ \text{ELBO}(x, x') &= \mathbb{E}_{z \sim q(z \mid x, x')} \left[\log p_{\text{edit}}(x \mid x', z)\right] - \text{KL}\left(q(z \mid x, x') \,\|\, p(z)\right) \end{aligned}$$

3 Optimization

- We will use stochastic gradient ascent to maximize the objective.
  1. Sample a sentence x uniformly from $\mathcal{X}$.
  2. Sample a prototype x' uniformly from $\mathcal{N}(x)$.
     - For speed, $\mathcal{N}(x)$ should be precomputed.
  3. Compute $g = (g_p, g_q)$, an unbiased estimate of $\nabla_\Theta \text{ELBO}(x, x')$ (see below for definitions of g_p and g_q)

(a) Sample an edit vector, $z \sim q(z \mid x, x')$:
   - Compute $f = f(x, x')$ as described in Section 3.4 of "Generating Sentences by Editing Prototypes".
   - Define $f_{\text{norm}} = \|f\|_2$ and $f_{\text{dir}} = f/f_{\text{norm}}$.
   - Define $\tilde{f}_{\text{norm}} = \min(f_{\text{norm}}, 10 - \epsilon)$.
   - Sample $z_{\text{dir}} \sim \text{vMF}(f_{\text{dir}}, \kappa)$.
     * This must be done using a reparameterization trick, which introduces:
       · A set of auxiliary random variables, $\alpha = (\omega, v)$
       · A deterministic function h, such that $z_{\text{dir}} = h(f_{\text{dir}}, \alpha)$
     * See the next section for details.
   - Sample $z_{\text{norm}} \sim \text{Unif}\left[\tilde{f}_{\text{norm}}, \tilde{f}_{\text{norm}} + \epsilon\right]$.
     * This is done using the following (very simple) reparameterization trick:
       · Sample auxiliary random variable $o \sim \text{Unif}[0, \epsilon]$
       · Define $z_{\text{norm}} = \tilde{f}_{\text{norm}} + o$
   - Define $z = z_{\text{dir}} \cdot z_{\text{norm}}$

(b) Compute $g_p = \nabla_{\Theta_p} \log p_{\text{edit}}(x \mid x', z)$
   - g_p is computed using standard backpropagation through the editor, treating x, x' and z as constants.

(c) Compute $g_q = \nabla_{\Theta_q} \log p_{\text{edit}}(x \mid x', z)$
   - g_q is computed using standard backpropagation through the editor ***as well as*** through $z_{\text{norm}} = \tilde{f}_{\text{norm}} + o$ and $z_{\text{dir}} = h(f_{\text{dir}}, \alpha)$, treating x, x', o and α as constants.
   - Note that z_{norm} and z_{dir} are ***not*** treated as constants, but instead as functions that we backpropagate through. See the next section for the functional form of h.

(d) Define $g = (g_p, g_q)$

4. Update parameters
   - $\Theta \leftarrow \Theta + \lambda g$ where λ is some learning rate.
   - Alternatively, this step could be replaced by a more sophisticated learning rule such as Adam, RMSprop, etc.

4 Sampling from a von-Mises Fisher distribution

- We would like to sample a vector $z_{\text{dir}} \in \mathbb{R}^p$ from $\text{vMF}(\mu, \kappa)$, a von-Mises Fisher distribution with direction $\mu \in \mathcal{S}^{p-1}$ (a point on the unit sphere in p-dimensional space) and concentration $\kappa \in \mathbb{R}$ (must be ≥ 0).

- We will introduce a set of auxiliary random variables, $\alpha = (\omega, v)$

  - ω is a random scalar, with distribution $p(\omega)$ defined as:

$$p(\omega) = \begin{cases} C \cdot e^{\kappa\omega} \left(1 - \omega^2\right)^{(p-3)/2} & \omega \in [-1, 1] \\ 0 & \text{otherwise} \end{cases}$$

    * $C = \left(\frac{\kappa}{2}\right)^{p/2-1} \left\{\Gamma\left(\frac{p-1}{2}\right) \Gamma\left(\frac{1}{2}\right) I_{(p-1)/2}(\kappa)\right\}^{-1}$ is a normalization constant.
    * Γ is the gamma function.
    * $I_n(\kappa)$ is the modified Bessel function of the first kind.
    * No exact method for sampling from $p(\omega)$ is currently known. See the next section for a rejection sampling strategy for sampling from $p(\omega)$.

- v is a random vector in $\mathbb{R}^{p-1}$ with distribution $p(v)$ defined to be the uniform distribution on the $(p-2)$ sphere, $\mathcal{S}^{p-2} = \left\{x \in \mathbb{R}^{p-1} \ : \ d(x, \mathbf{0}) = 1\right\}$.
  - * This can be sampled by simply drawing a multivariate normal random vector and normalizing it to length 1, but there are other more efficient approaches.

- Define $p(\alpha) = p(\omega)\, p(v)$ (implying that ω and v are independent)
- We can now sample $z_{\text{dir}} \sim \text{vMF}(\mu, \kappa)$ as follows:

  1. Sample $\omega \sim p(\omega)$
  2. Sample $v \sim p(v)$
  3. Define $s = \left(\omega;\ v^\top \cdot \sqrt{1-\omega^2}\right)^\top$
  4. Construct a Householder reflection matrix, R
     - Let $e_1 = \begin{bmatrix}1 & 0 & 0 & \ldots\end{bmatrix}$
     - Define $r = (e_1 - \mu) / \|e_1 - \mu\|$
     - Let $R = I - 2rr^\top$, where I is the identity matrix

  - Define $z_{\text{dir}} = Rs$
    - * R essentially reflects s across the hyperplane that lies between μ and e_1

- For the sake of clarity, we can also write these steps in a form that more clearly illustrates how z_{dir} is a function of μ and α:

$$\begin{aligned}\alpha &\sim p(\alpha)\\ z_{\text{dir}} &= h(\mu, \alpha) = \left(I - 2\left[(e_1-\mu)/\|e_1-\mu\|\right]\left[(e_1-\mu)/\|e_1-\mu\|\right]^\top\right)\left(\omega;\ v^\top \cdot \sqrt{1-\omega^2}\right)^\top\end{aligned}$$

5 Sampling $p(\omega)$ using rejection sampling

- To draw a sample ω from $p(\omega)$, we will utilize the following rejection sampling algorithm:

  1. Define $a = \frac{(p-1)+2\kappa+\sqrt{4\kappa^2+(p-1)^2}}{4}$
  2. Define $b = \frac{-2\kappa+\sqrt{4\kappa^2+(p-1)^2}}{p-1}$
  3. Define $d = \frac{4ab}{1+b} - (p-1)\ln(p-1)$
  4. Repeat until acceptance criterion is satisfied
     (a) Sample $\beta \sim \text{Beta}\left(\frac{p-1}{2}, \frac{p-1}{2}\right)$
     (b) Propose $\omega = \frac{1-(1+b)\beta}{1-(1-b)\beta}$
     (c) Define $t = \frac{2ab}{1-(1-b)\beta}$, and sample $u \sim \text{Unif}[0,1]$
     (d) If $(p-1)\ln(t) - t + d \geq \ln(u)$, accept. Otherwise, start over.

- Note:
  - This rejection sampling algorithm comes from Davidson 2018.
  - Davidson 2018 uses the algorithm of Ulrich 1984, but corrects two typos that existed in the original algorithm (Algorithm VM):
    - * The proposal for ω was incorrectly defined to be $\omega = \frac{1-(1+b)\beta}{1+(1-b)\beta}$
    - * t was incorrectly defined to be $t = \frac{2ab}{1+(1-b)\beta}$
  - For an alternative method of sampling ω, see Wood 1994.

6 References

- Hyperspherical Variational Auto-encoders (Davidson et al 2018)
  - Uses Ulrich's approach, but corrects two typos.
- Directional Statistics (Mardia and Jupp 1999)
  - page 172, Section 9.3.2, "Simulation"
  - Does not give the algorithm for sampling ω
  - Method of combining v and ω appears to be wrong: in particular, v is the wrong dimension (p rather than $p-1$), and v and ω are combined incorrectly (addition rather than concatenation)
- Math Stack Exchange
  - Claims to be the Ulrich-Wood algorithm, but the implementation is incorrect: appears to make the same mistake made in "Directional Statistics" (Mardia and Jupp 1999)
- Simulation of the von Mises Fisher distribution (Wood 1994)
  - Behind a paywall
  - Points out that there are errors in the original Ulrich 1984 paper
  - Proposes a different rejection sampling scheme
- Ulrich 1984
  - The original paper on sampling from a von Mises Fisher distribution
  - Contains two typos in the sampling algorithm