David Yenicelik

ETH Zürich

eth logo

A dissertation submitted to ETH Zürich
in partial fulfilment of the requirements for the degree of
Master of Science ETH in Computer Science

Eidgenössische Technische Hochschule Zürich
Data Analytics Lab
Institute of Machine Learning
CAB F 42.1, Universitätsstrasse 6, 8006, Zürich
Zürich 8006
Switzerland

Email: yedavid@ethz.ch

November 15, 2019

Declaration

I David Yenicelik of ETH Zürich, being a candidate for the M.Sc. in Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 14,235

Signed:

Date:

Abstract

Write a summary of the whole thing. Make sure it fits in one page.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In Natural Language Processing (NLP), bilingual lexical induction (BLI) is a problem of inferring word-to-word mapping between two languages. While supervised BLI may be learned trivially from a dictionary, unsupervised BLI is highly non-trivial, and serves as a backbone to many unsupervised Neural Machine Translation (NMT) systems, without which the overall MT performance drastically drops [?] [?]. In addition to the unsupervised setting, words in a source language A often do not carry a one-to-one correspondence with words in a target language B. This further increases the difficulty of finding a (bijective) mapping between the two embedding spaces.

1.1 Scope of Work

Continuing where [?] left off, we want to investigate the performance of using normalising flows to model unsupervised lexicon matching between two probability distributions, which are defined by Gaussian emebddings, which have a one-to-one correspondence to tokens in the respective languages. We aim to use Gaussian embeddings for the robustness, and better integration into the probabilistic perspective. The method discussed in the paper also relies a lot on the (semi-)supervised loss component. We aim to investigate why this is the case, and would like to revise a method which is more robust in the fully unsupervised setting.

Minimal goals: We propose the following steps on achieving this goal.

1. Replicate the algorithms and models which can generate through Gaussian embedding [?]. Do one sanity check by doing a sanity check on one of (SimLex / WordSim)
2. Replicate "Density matching for bilingual word embedding" to setup a baseline normalising flow between vector-word-embeddings [?].
3. Define loss between predicted and target embeddings (if change in definition is necessary)
4. Change the embeddings in point 2. to use Gaussian embeddings. Implement Loss functions found in point 3.

Extended goals: If the above points provide good performance , we would like to expand on the below points.

1. Implement a fully unsupervised extension by investigating the shortcomings of [?].
2. Investigate "deeper" normalising flows than the linear flow in [?] as such [?] [?].

Contingency Plan / Further extended goals: Implementing the following points would allow for an applied perspective of this approach, showing that this methodology allows for more robust mapping, also outside the field of NLP.

1. Train embeddings for job-systems ESCO and AUGOV using Skip-Gram or Co-Occurence-matrix based. Do a sanity check.
2. Train Gaussian embeddings for job-systems. Do a sanity check.
3. Generate a small validation dataset between the European- and Australian job-system.

4. Setup some baseline algorithms based on NLP, graph-matching, colinear-PCA for matching as a non-NLP benchmark environment. Compare against above-proposed methods.
5. Find a normlising flow model to transform one job system into another.

Chapter 2

Background

We provide a short introduction to the background work.

2.1 Gaussian Embeddings

The above two problems can be regarded as finding an invertible transformation from one (embedding) space $\mathcal{X} \in \mathbf{R}^d$ into another $\hat{\mathcal{X}} \in \mathbf{R}^d$. Normalising flows [**?**] [**?**] have proven to be a powerful tool in modeling such relations. As such, the aim of this project is to find a model based on normalising flows which is able to find an an invertible mapping f from $\mathcal{X}$ to $\hat{\mathcal{X}}$. To keep the discussion focused, this thesis deals with the problem of finding a model for bilingual lexicon matching.

Gaussian Embeddings

2.2 Normalising Flows

(Rezende 2015) A simple distribution is transformed into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained.

In the above equation, the KL-divergence is the loss between the approximate posterior and the prior distribution (which acts as a regualizer). The second term, which is the expected log-likelihood is the reconstruction error.

Inference with normalizing flows provides a tighter, modified variational lower bound with additional terms that only add terms with linear time complexity.

in the asymptotic regine, this is able to recover the true posterioir distribution

Approximate posterior distribution of the latent variables $q_\phi(z|x)$

$$\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x|z)p(z)dz && (2.1)\\
&= \log \int \frac{q_\theta(z|x)}{q_\theta(z|x)} p_\theta(x|z)p(z)dz && (2.2)\\
&= \mathbb{D}_{KL}\left[q_\phi(z|x)||p(z)\right] + \mathbb{E}_q\left[\log p_\theta(x|z)\right] && (2.3)\\
&\geq -\mathcal{F}(x) && (2.4)
\end{aligned}$$

where $-\mathbf{F}(x)$ is a free energy function, and where we used Jensen's inequality to obtain the final equation (through $logE_q\left[\frac{p(x,z)}{q(z)}\right] \geq logE_q\left[\log\frac{p(x,z)}{q(z)}\right]$). $p_\theta(x|z)$ is a likelihood function and $p(z)$ is a prior over latent variables, and $q_\theta(z|x)$ is a (simple) model distribution with which we want to approximate $p(x|z)$. item We try to minimize this loss

To train the model, we need to efficiently (1) compute the derivatives of the expected log-likelihood $\nabla\phi\mathbb{E}_{q_\phi(z)}\left[p_\theta(x|z)\right]$ and (2) choose the richest, computationally-feasible approximate posterior distribution $q(\dot{})$.

Using Monte Carlo gradient estimation and inference networks, which (when used together), they call *amortized variational inference.* TODO: TIMO SAID THAT THIS IS WRONGLY APPROXIMATING THE TRUE DIS-

TRIBUTION? WHAT ABOUT PSI?

For stochastic backpropagation, we make use of two techniques:

- Reparameterization: The latent variable is reparametrized in terms of a known base distribution and a differentiable transofmration. As an example, if $q_\phi(z)$ is a Gaussian distribution $\mathbf{N}(z|\mu, \sigma^2)$ with trainable parameters $\phi = \mu, \sigma^2$, then we can reparametrize the variable z as

$$z \sim \mathbf{N}(z|\mu, \sigma^2) \iff z = \sigma + \sigma\epsilon, \epsilon \sim \mathbf{N}(0, 1) \tag{2.5}$$

- Backpropagation with Monte Carlo. We backpropagated w.r.t. the parameters ϕ of the variational distribution using a Monte Carlo aprxoimation. This is our samples batch.

$$\nabla\phi\mathbb{E}_{q_\phi(z)}\left[f_\theta(z)\right] \iff \nabla\phi\mathbb{E}_{\mathbf{N}(\epsilon|0,1)}\left[\nabla_\phi f_\theta(\mu + \sigma\epsilon)\right] \tag{2.6}$$

  where we have simply rewritten z using the reparametrization trick.

For continuous latent variables, it has the lowest variance among competing estimators [CITEEE].

Normalizing flows are most often used in the context of inference network. An inference network learns an inverse map from observations to latent variables. The simplest latent variable is a Gaussian distribution.

Rezende 2015 propose a *Deep Latent Gaussian Model*, which consists of a hierarchy of L layers of Gaussian latent variables z_l for layer l.

$$p(x, z_1, \ldots, z_L) = p(x|f_0(z_1)) \prod_{l=1}^{L} p(z_l|f_l(z_{l+1})) \tag{2.7}$$

We induce priors over each latent variable $p(z_l) = \mathbf{N}(0, I)$, and the observatoin distribution $p_\theta(x|z)$ is any distribution that is conditioned on z_1 and is parametrized by a neural network.

This model is very general, and captures includes other models, such as factor analysis and PCA; non-linear factor analyiss, and non-linear Gaussian belif networks as special cases (Rezende 2014, cited in Rezende 2015)

We know that $\mathbf{D}_{KL}\left[q||p\right] = 0$ is achived when $q_\phi(z|x) = p_\theta(z|x)$ (i.e. the approximate q matches the true posterior distribution). IS THIS THE ONLY TIME THIS CAN BE ACHIEVED?

A normalising flow captures more flexibel distributions.

Summary normalising flows:

A normalizing flow is a set of basic rules for transofmration of densiities, considers an invertible, smooth mapping

$$f : \mathbf{R}^d \rightarrow \mathbf{R}^d$$

where the input dimensioinality d_0 and output dimensionalities d_L match. The inverse is denoted $f^{-1} = g$, i.e. the composition $g \circ f(z) = z$. When we want to transform a random variable $z' = f(z)$, we receive the following distribution:

This stems from the Identity that the probability mass must be conversed amongst operations:

$$p_x(x) = \frac{p_z(z)V_Z}{V_X} \tag{2.8}$$

$$z' = f(z) \tag{2.9}$$

$$\frac{z'}{q(z')} = \frac{f(z)}{q(z)} \tag{2.10}$$

WHERE DOES THE PRIME COME FROM

We can then construct arbitrarily complex densities by composing several simple maps and successively applying the above mass-preserving operation.

$$z_K = f_K \circ \ldots \circ f_2 \circ f_1(z_0) \tag{2.11}$$

We then apply the log-function to have tractable probability functions.

Probability mass must be conserved. From the change of variable formula, we know that taking infinitesimal c:hanges toward the direction of the final probabiliity distribution, we get

$$\int z' q(z') dz' = \int f(z) q(z) dz \tag{2.12}$$

$$q(z') = q(z) \ln \left| det \frac{\delta f(z)}{\delta z'} \right| \tag{2.13}$$

There are different ways to use normalising flows:

Although computing the above gradient is expensive ($O(n^3)$ complexity), we can use the Matrix determinant lemma, and construct out mapping matrices in a special way as follows (https://blog.evjang.com/2018/01/nf1.htm)

$$
\begin{aligned}
det(A) &= det(L + V\dot{D}\dot{V}^T) && (2.14)\\
&= det(L)(1 + (VD^{\frac{1}{2}})^T(L^{-1}D^{\frac{1}{2}}V)) && (2.15)
\end{aligned}
$$

where we have used the Matrix Determiinant Lemma for the second equality, with L is a lower triangular matrix, and D is a diagonal matrix.

subsectionExtensions on Normalising Flows

(https://www.shakirm.com/slides/DeepGenModelsTutorial.pdf)

Ever since Normalising flows were proposed, a number of modified versions were presented by [CITE].

Each one calculates the consecutive flow based on a different composition of the previous latent variables z_k.

Planar Flow

The planar flow is a simple sequential flow with invertible piecewise linear transformations.

$$z_k = z_{k-1} + uh(w^T z_{k-1} + b) \tag{2.16}$$

Real NVP

The input for each step is split into two parts. The intuition behind this is similar to a residual network, which includes logic that allows the gradient to flow back easier.

$$
\begin{aligned}
y_{1:d} &= z_{k-1,1:d} && (2.17)\\
y_{d+1:D} &= t(z_{k-1,1:d}) + z_{d+1:D} \odot \exp(s(z_{k-1,1:d})) && (2.18)
\end{aligned}
$$

Here, t and s are arbitrary functions, such as deep neural networks or simple linear transforms.

Inverse AR Flow

At each timestep, all prior variables are taken into consideration.

$$z_k = \frac{z_{k-1} - \mu_k(z_{<k}, x)}{\sigma_k(z_{<k}, x)} \tag{2.19}$$

Non-Linear Independent Components Estimation (NICE)

Another volume-preserving flow is the NICE flow. First, we arbitrarily partition the latent vector into two components $z = (z_A, z_B)$.

$$f(z) = (z_A, z_B + h_\lambda(z_A)) \tag{2.20}$$

$$g(z') = (z'_A, z'_B + h_\lambda(z'_A)) \tag{2.21}$$

Here, h_λ can be chosen in such a way that h is a deep neural network with parameters λ

LOL, COULD WE JUST CREATE A NEW ONE WITH MORE PROPERTIES??

Multiple

Normalising flows [**?**], [**?**] are a statistical technique where a series of invertible transformations f_t are applied to a simple distribution $z_0 \sim q_0(z)$, to yield increasingly complex distributions $z_t = f_t(z_{t-1})$, s.t. the last iterate z_T has the desired and more flexible distribution. As long as we can efficiently compute the Jacobian determinant of the transformation bijection f_t, we can both (1) evaluate the density of our data (by applying an inverse transformation and computing the density in the base distribution), and (2) sample from our complex distribution (by sampling from the base distribution and applying the forward transformation) These can be used for classification and clustering [?], [variational inference tasks [?] such as image-generation [?]], enriching the posterior (and prior!) [?], and density estimation [?].

Glow: Generative Flows with Invertible 1x1 Convolutions

We use 1x1 convolutional layers as a generalization of the permutation in sequential flows with lower triangular learnable rotation matrices.

TODO: Convolution operators and reduction to permutation operators

$$\log \left| \mathrm{def} \left(\frac{d\mathrm{conv2D}(h; W)}{dh} \right) \right| = hw\log =$$

(read the paper a bit more before citing this, but cite this: https://arxiv.org/pdf/1901.08624.p

Flow++

The coupling layers can be described as

$$y_1 = x_1 y_2 \qquad = x_2 \dot{e}xp(a_\theta(x_1)) + b_\theta(x_1) \tag{2.23}$$

where the functions a_θ and b_θ are learnable functions (possibly normalizing flows which). These have to be invertible affine transformations

Chapter 3

Related Work

3.1 Unsupervised Biliingual Lexicon Matching

Chapter 4

Analysis of the current state of the art

Chapter 5

Our Method

Chapter 6

Evaluation

Chapter 7

Conclusion