Riemannian Normalizing Flow on Variational Wasserstein Autoencoder for Text Modeling

Prince Zizhuang Wang
Department of Computer Science
University of California, Santa Barbara
zizhuang_wang@ucsb.edu

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
william@cs.ucsb.edu

Abstract

Recurrent Variational Autoencoder has been widely used for language modeling and text generation tasks. These models often face a difficult optimization problem, also known as the Kullback-Leibler (KL) term vanishing issue, where the posterior easily collapses to the prior, and the model will ignore latent codes in generative tasks. To address this problem, we introduce an improved Wasserstein Variational Autoencoder (WAE) with Riemannian Normalizing Flow (RNF) for text modeling. The RNF transforms a latent variable into a space that respects the geometric characteristics of input space, which makes posterior impossible to collapse to the non-informative prior. The Wasserstein objective minimizes the distance between the marginal distribution and the prior directly, and therefore does not force the posterior to match the prior. Empirical experiments show that our model avoids KL vanishing over a range of datasets and has better performances in tasks such as language modeling, likelihood approximation, and text generation. Through a series of experiments and analysis over latent space, we show that our model learns latent distributions that respect latent space geometry and is able to generate sentences that are more diverse. [1]

1 Introduction

Variational Autocoder (VAE) (Kingma and Welling, 2013; Rezende and Mohamed, 2015) is a probabilistic generative model shown to be successful over a wide range of tasks such as image generation (Gregor et al., 2015; Yan et al., 2016), dialogue generation (Zhao et al., 2017b), transfer learning (Shen et al., 2017), and classification (Jang et al., 2017). The encoder-decoder architecture of VAE allows it to learn a continuous space of latent representations from high-dimensional data input and makes sampling procedure from such latent space very straightforward. Recent studies also show that VAE learns meaningful representations that encode non-trivial information from input (Gao et al., 2018; Zhao et al., 2017a).

Applications of VAE in tasks of Natural Language Processing (Bowman et al., 2015; Zhao et al., 2017b, 2018; Miao et al., 2016) is not as successful as those in Computer Vision. With *long-short-term-memory* network (LSTM) (Hochreiter and Schmidhuber, 1997) used as encoder-decoder model, the recurrent variational autoencoder (Bowman et al., 2015) is the first approach that applies VAE to language modeling tasks. They observe that LSTM decoder in VAE often generates texts without making use of latent representations, rendering the learned codes as useless. This phenomenon is caused by an optimization problem called KL-divergence vanishing when training VAE for text data, where the KL-divergence term in VAE objective collapses to zero. This makes the learned representations meaningless as zero KL-divergence indicates that the latent codes are independent of input texts.

Many recent studies are proposed to address this key issue. Yang et al. (2017); Semeniuta et al. (2017) use *convolutional neural network* as decoder architecture to limit the expressiveness of decoder model. Xu and Durrett (2018); Zhao et al. (2017a,b, 2018) seek to learn different latent space and modify the learning objective. And, even though not designed to tackle KL vanishing at the beginning, recent studies on *Normalizing Flows* (Rezende and Mohamed, 2015; van den Berg et al., 2018) learn meaningful latent space as it helps to transform an over-simplified latent distribution into more flexible distributions.

In this paper, we propose a new type of flow,

[1] Code could be found at https://github.com/kingofspace0wzz/wae-rnf-lm

Proceedings of NAACL-HLT 2019, pages 284–294
Minneapolis, Minnesota, June 2 - June 7, 2019.

called Riemannian Normalizing Flow (RNF), together with the recently developed Wasserstein objective (Tolstikhin et al., 2018; Arjovsky et al., 2017), to ensure VAE models more robust against the KL vanishing problem. As further explained in later sections, the Wasserstein objective helps to alleviate KL vanishing as it only minimizes the distance between latent marginal distribution and the prior. Moreover, we suspect that the problem also comes from the over-simplified prior assumption about latent space. In most cases, the prior is assumed to be a standard Gaussian, and the posterior is assumed to be a diagonal Gaussian for computational efficiency. These assumptions, however, are not suitable to encode intrinsic characteristics of input into latent codes as in reality the latent space is likely to be far more complex than a diagonal Gaussian.

The RNF model we proposed in this paper thus helps the situation by encouraging the model to learn a latent space that encodes some geometric properties of input space with a well-defined geometric metric called Riemannian metric tensor. This renders the KL vanishing problem as impossible since a latent distribution that respects input space geometry would only collapse to a standard Gaussian when the input also follows a standard Gaussian, which is never the case for texts and sentences datasets. We then empirically evaluate our RNF Variational Wasserstein Autoencoder on standard language modeling datasets and show that our model has achieved state-of-the-art performances. Our major contributions can be summarized as the following:

- We propose Riemannian Normalizing Flow, a new type of flow that uses the Riemannian metric to encourage latent codes to respect geometric characteristics of input space.
- We introduce a new Wasserstein objective for text modeling, which alleviates KL divergence term vanishing issue, and makes the computation of normalizing flow easier.
- Empirical studies show that our model produces state-of-the-art results in language modeling and is able to generate meaningful text sentences that are more diverse.

2 Related Work

2.1 Variational Autoencoder

Given a set of data $\mathbf{x} = (x_1, x_2, ..., x_n)$, a Variational Autoencoder (Kingma and Welling, 2013) aims at learning a continuous latent variable that maximizes the log-likelihood $\log p(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$. Since this marginal is often intractable, a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is used to approximate the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. VAE tries to maximize the following lower bound of likelihood,

$$\mathcal{L}(\theta; \phi; \mathbf{x}) = \mathbb{E}_{q(\mathbf{x})}[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \quad (1)$$

$$- KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))] \quad (2)$$

where $q(\mathbf{x})$ is the empirical distribution of input, and the prior $p(\mathbf{z})$ is often assumed to be a standard Gaussian for simplicity. The first term in the objective is the reconstruction error, and the second one is KL divergence. For modeling text sentences, Bowman et al. (2015) parameterizes both the inference model $q(\mathbf{z}|\mathbf{x})$ and the generative model $p(\mathbf{x}|\mathbf{z})$ as LSTMs. The reparameterization trick proposed by Rezende and Mohamed (2015) is used to train these two models jointly.

2.2 KL Divergence Term Vanishing

Since the generative model is often an LSTM that has strong expressiveness, the reconstruction term in the objective will dominate KL divergence term. In this case, the model is able to generate texts without making effective use of latent codes as the latent variable $\mathbf{z}$ becomes independent from input when KL divergence term collapses to zero.

There are two main approaches to address this issue. One is to explore different choices of the decoder model to control the expressiveness of LSTM. Yang et al. (2017) and Semeniuta et al. (2017) use CNN as an alternative to LSTM. The dilation technique used by Yang et al. (2017) also helps to control the trade-off between decoder capacity and KL vanishing. The other approach is to change the form of latent distribution and to modify the training objective. Xu and Durrett (2018) proposes to use hyperspherical distribution and shows that the KL vanishing problem does not happen in hypersphere. The infoVAE (Zhao et al., 2017a) argues that VAE objective is anti-informatics which encourages KL divergence to be zero. They, therefore, add a mutual information term $I(\mathbf{z}; \mathbf{x})$ explicitly to ensure that latent variable $\mathbf{z}$ encodes non-trivial information about $\mathbf{x}$, in

which case the KL would be greater than zero as $\mathbf{z}$ is no longer independent from $\mathbf{x}$. In a similar manner, Xiao et al. (2018) introduces a Dirichlet latent variable to force latent codes to learn useful topic information given input documents. He et al. (2019) and Kim et al. (2018) achieve the current state-of-the-art in terms of sample perplexity. Moreover, the bag-of-words loss used by Zhao et al. (2017b) also dramatically alleviates KL vanishing while not sacrificing sample quality. In Section 3 and Section 4, we introduce RNF with Wasserstein objective. Our proposed model also lies in the direction which seeks to learn more flexible latent distribution, as the main advantage of RNF is to ensure a flexible latent space able to capture geometric characteristics of input space.

3 Background

In this section, we review the basic concepts of Riemannian geometry, normalizing flow, and Wasserstein Autoencoder. We then introduce our new Riemannian normalizing flow in Section 4.

3.1 Riemannian Geometry Review

Consider an input space $\mathcal{X} \subset R^D$, a d-dimensional $(d < D)$ manifold is a smooth surface of points embedded in $\mathcal{X}$. Given a manifold $\mathcal{M}$, a Riemannian manifold is a metric space $(\mathcal{M}, G)$, where G is the Riemannian metric tensor that assigns an inner product to every point on the manifold. More formally, a Riemannian metric $G : \mathcal{Z} \to R^{d\times d}$ is defined as a smooth function such that for any two vectors u, v in the tangent space $T_z M$ of each point $z \in \mathcal{M}$, it assigns the following inner product for u and v,

$$< u, v >_G = u^T G(z) v \tag{3}$$

The Riemannian metric helps us to characterize many intrinsic properties of a manifold. Consider an arbitrary smooth curve $\gamma(t) : [a, b] \to \mathcal{M}$ on a given manifold $\mathcal{M}$ with a Riemannian metric tensor G, the length of this curve is given by

$$\begin{aligned}\mathcal{L}(\gamma) &= \int_a^b ||\gamma'(t)|| dt \\ &= \int_a^b \sqrt{< \gamma'_t, \gamma'_t >_G} dt \\ &= \int_a^b \sqrt{\gamma'^T_t G(\gamma_t)\gamma'_t} dt \end{aligned} \tag{4}$$

where γ'_t is the curve velocity and lies in the tangent space $T_{\gamma_t} M$ at point $\gamma(t)$. When the metric tensor G is equal to 1 everywhere on the curve, it becomes a metric tensor on Euclidean space, where the length of curve is defined as the integral of the velocity function, $\mathcal{L}(\gamma) = \int_a^b \sqrt{\gamma'^T_t \gamma'_t} dt = \int_a^b \gamma'_t dt$. Given the definition of curve length, the geodesic path between any two points can be defined as the curve that minimizes the curve length γ. Namely, if γ_t is the geodesic curve connecting $\gamma(a)$ and $\gamma(b)$, then

$$\gamma_t = \underset{\gamma}{\operatorname{argmin}}\, \mathcal{L}(\gamma) \tag{5}$$

Practically, a geodesic line is often found by optimizing the following energy function,

$$\begin{aligned} E(\gamma) &= \frac{1}{2} \int_a^b \gamma'^T_t G(\gamma_t) \gamma'_t dt \\ \gamma_t &= \underset{\gamma}{\operatorname{argmin}}\, E(\gamma) \end{aligned} \tag{6}$$

Note that the Euclidean metric is a special case of Riemannian metric. The more general metric tensor G gives us a sense of how much Riemannian geometry deviates from Euclidean geometry.

3.2 Review: Normalizing Flow

The powerful inference model of VAE can approximate the true posterior distribution through variational inference. The choice of this approximated posterior is one of the major problems. For computational efficiency, a diagonal Gaussian distribution is often chosen as the form of the posterior. As the covariance matrix is always assumed to be diagonal, the posterior fails to capture dependencies among individual dimensions of latent codes. This poses a difficult problem in variational inference. As it is unlikely that the true posterior has a diagonal form, the approximated diagonal distribution is not flexible enough to match the true posterior even in asymptotic time.

A normalizing flow, developed by (Rezende and Mohamed, 2015), is then introduced to transform a simple posterior to a more flexible distribution. Formally, a series of normalizing flows is a set of invertible, smooth transformations $f_t : R^d \to R^d$, for $t = 1, ..., T$, such that given a random variable z_0 with distribution $q(z_0)$, the resulting random variable $z^T = (f_T \circ f_{T-1} \circ ... \circ f_1)(z_0)$ has the following density function,

$$q(z_T) = q(z_0) \prod_{t=1}^{T} |det \frac{\partial f_t^{-1}}{\partial z_{t-1}}| \tag{7}$$

Since each transformation f_i for $i = 1, ..., T$ is invertible, its Jacobian determinant exists and can be computed. By optimizing the modified *evidence lower bound* objective,

$$\ln p(x) \geq \mathbb{E}_{q(z_0|x)} \left[\ln p(x|z_T) + \sum_{t=1}^{T} \ln |det \frac{\partial f_t}{\partial z_{t-1}}| \right] - KL(q(z_0|x)||p(z_T)) \quad (8)$$

the resulting latent codes z_T will have a more flexible distribution.

Based on how the Jacobian-determinant is computed, there are two main families of normalizing flow (Tomczak and Welling, 2016; Berg et al., 2018): *general normalizing flow* and *volume preserving flow*. While they both search for flexible transformation that has easy-to-compute Jacobian-determinant, the volume-preserving flow aims at finding a specific flow whose Jacobian-determinant equals 1, which simplifies the optimization problem in equation (6). Since we want a normalizing flow that not only gives flexible posterior but also able to uncover the true geometric properties of latent space, we only consider general normalizing flow whose Jacobian-determinant is not a constant as we need it to model the Riemannian metric introduced earlier.

3.3 Wasserstein Autoencoder

Wasserstien distance has been brought to generative models and is shown to be successful in many image generation tasks (Tolstikhin et al., 2018; Arjovsky et al., 2017; Bousquet et al., 2017). Instead of maximizing the *evidence lower bound* as VAE does, the Wasserstein Autoencoder (Tolstikhin et al., 2018) optimizes the optimal transport cost (Villani, 2008) between the true data distribution $P_X(x)$ and the generative distribution $P_G(x)$. This leads to the Wasserstein objective,

$$D(P_X, P_G) = \inf_{Q(Z|X) \in \mathcal{Q}} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)}[c(X, G(Z))] + \lambda D_Z(Q_Z, P_Z) \quad (9)$$

where $c(\cdot)$ is the optimal transport cost, $G : \mathcal{Z} \to \mathcal{X}$ is any generative function, and the coefficient λ controls the strength of regularization term D_Z. Given a positive-definite reproducing kernel $k : \mathcal{Z} \times \mathcal{Z} \to \mathcal{R}$, the regularization term D_Z can be approximated by the *Maximum Mean Discrepancy* (**MMD**) (Gretton et al., 2012) between the prior P_Z and the aggregate posterior $Q_Z(z) = \int q(z|x)p(x)dx$,

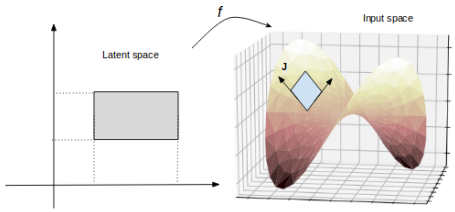

Figure 1: Parameterization of the input manifold by latent space and generative function f. J is the Jacobian tangent space at a point on the manifold, which reflects how curved the neighborhood is around that point.

$$MMD_k(P_Z, Q_Z) = || \int_{\mathcal{Z}} k(z, \cdot) dP_Z - \int_{\mathcal{Z}} k(z, \cdot) dQ_Z || \quad (10)$$

4 Our Approach

In this section we propose our *Riemannian Normalizing Flow* (**RNF**). RNF is a new type of flow that makes use of the Riemannian metric tensor introduced earlier in Section 3. This metric enforces stochastic encoder to learn a richer class of approximated posterior distribution in order to follow the true geometry of latent space, which helps to avoid the local optimum in which posterior collapses to a standard prior. We then combine this with WAE and we will explain why and how WAE should be used to train with RNF.

4.1 Riemannian Normalizing Flow

In the context of VAE, learning a latent space that is homeomorphic to input space is often very challenging. Consider a manifold $\mathcal{M} \subset R^D$, a generator model $x = f(z) : \mathcal{Z} \to R^D$ serves as a low-dimensional parameterization of manifold $\mathcal{M}$ with respect to $z \in \mathcal{Z}$. For most cases, latent space $\mathcal{Z}$ is unlikely to be homeomorphic to $\mathcal{M}$, which means that there is no invertible mapping between $\mathcal{M}$ and $\mathcal{Z}$. And, since the inference model $h : \mathcal{M} \to \mathcal{Z}$ is nonlinear, the learned latent space often gives a distorted view of input space. Consider the case in Figure 2, where the leftmost graph is the input manifold, and the rightmost graph is the corresponding latent space with curvature reflected by brightness. Let us take two arbitrary points on the manifold and search for the geodesic path connecting these two points. If we consider the distorted latent space as Euclidean, then the geodesic path in latent space does not reflect the true shortest distance between these two points on the manifold, as a straight line in the latent space would

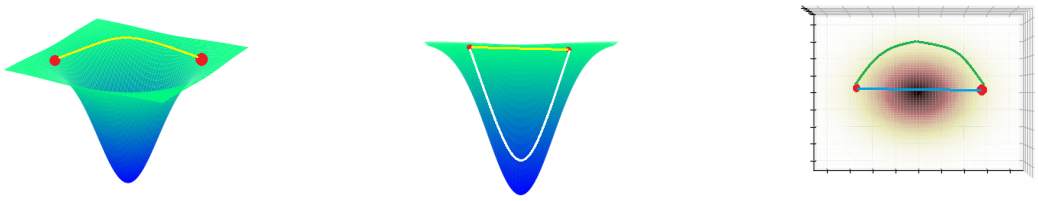

Figure 2: An example when latent space does not reflect input space. *Left:* a manifold that is highly curved in the central region. The yellow line is the geodesic (shortest) path connecting two sample points shown on the manifold. *Right:* The projection of manifold into 2D latent space, where the color brightness indicates curvature with respect to the manifold. The green line is the geodesic path if taking the curvature into account, while the blue line is the geodesic path if we regard latent space as Euclidean. *Middle:* The corresponding geodesic paths projected back from latent space to manifold. The white line corresponds to the straight geodesic path in Euclidean space. It is far longer than the true geodesic on manifold since it does not take the curvature into account in latent space.

cross the whole manifold, while the true geodesic path should circumvent this hole. This distortion is caused by the non-constant curvature of latent space. Hence, the latent space should be considered as a curved space with curvature reflected by the Riemannian metric defined locally around each point. As indicated by the brightness, we see that the central area of latent space is highly curved, and thus has higher energy. The geodesic path connecting the two latent codes minimizes the energy function $E(\gamma) = \frac{1}{2}\int \gamma_t'^T G(\gamma_t)\gamma_t' dt$, indicating that it should avoid those regions with high curvature G.

The question now becomes how to impose this intrinsic metric and curvature into latent space. In this paper, we propose a new form of normalizing flow to incorporate with this geometric characteristic. First, consider a normalizing flow $f : \mathcal{Z} \to \mathcal{Z}'$, we can compute length of a curve in the transformed latent space $\mathcal{Z}'$,

$$\begin{aligned}\mathcal{L}(f(\gamma_t)) &= \int_a^b ||\mathbf{J}_{\gamma_t}\gamma_t'||dt \\ &= \int_a^b \sqrt{\gamma_t'\mathbf{J}_{\gamma_t}^T\mathbf{J}_{\gamma_t}\gamma_t'}dt \\ &= \int_a^b \sqrt{\gamma_t' G(\gamma_t)\gamma_t'}dt \end{aligned} \tag{11}$$

where

$$\begin{aligned}\gamma_t : [a,b] \to \mathcal{Z}', \qquad a, b \in \mathcal{Z} \\ \mathbf{J}_{\gamma_t} = \frac{\partial f}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\gamma_t} \\ G(\gamma_t) = \mathbf{J}_{\gamma_t}^T\mathbf{J}_{\gamma_t} \\ = (\frac{\partial f}{\partial \mathbf{z}})^T(\frac{\partial f}{\partial \mathbf{z}})\Big|_{\mathbf{z}=\gamma_t}\end{aligned} \tag{12}$$

$\mathbf{J}_{\gamma_t}$ is the Jacobian matrix defined at γ_t. In our case, the Riemannian metric tensor G is the inner product of Jacobian $\mathbf{J}_{\gamma_t}$ and is therefore symmetric positive definite. It reflects input space curvature in low-dimensional parameterization $\mathcal{Z}'$. In a highly curved region, the metric tensor $G = \mathbf{J}^T\mathbf{J}$ is larger than those in other areas, indicating that the latent representation of input manifold has lower curvature, or area of low energy, as any geodesic connecting each pair of points on the manifold favors lower energy path. This implies that those regions outside of data manifold should have high curvature reflected in their low-dimensional parameterization $\mathcal{Z}'$.

In this paper, we introduce *Riemannian normalizing flow* (**RNF**) to model curvature. For simplicity, we build our model based on *planar flow* (Rezende and Mohamed, 2015). A planar flow is an invertible transformation that retracts and extends the support of original latent space with respect to a plane. Mathematically, a planar flow $f : \mathcal{Z} \to \mathcal{Z}'$ has the form,

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^\mathbf{T}\mathbf{z} + b) \tag{13}$$

where $h : R^d \to R^d$ is any smooth non-linear function and is often chosen as $tanh(\cdot)$. The invertibiliy condition is satisfied as long as $\mathbf{u}^\mathbf{T}\mathbf{w} \geq -1$. Its Jacobian-determinant with respect to latent codes $\mathbf{z}$ is very each to compute,

$$\begin{aligned}|det\frac{\partial f}{\partial \mathbf{z}}| = |1 + \mathbf{u}^\mathbf{T}\phi(\mathbf{z})\mathbf{w}| \\ \phi(\mathbf{z}) = h'(\mathbf{w}^\mathbf{T}z + b)\end{aligned} \tag{14}$$

With the Jacobian-determinant of planar flow, it is straightforward to compute the determinant of metric tensor G. To see that, note that since $\frac{\partial f}{\partial \mathbf{z}}$:

$R^d \rightarrow R^d$ is a square matrix with full column rank due to invertibility of f, we have

$$|det\ G| = \left|\frac{\partial f}{\partial \mathbf{z}}^T\right|\left|\frac{\partial f}{\partial \mathbf{z}}\right| = \left|\frac{\partial f}{\partial \mathbf{z}}\right|^2 \quad (15)$$

To ensure well-behaved geometry in a transformed latent space, we need the Jacobian-determinant $|\frac{\partial f}{\partial \mathbf{z}}|$ to be large in region with high curvature $|det\ G|$. Hence, we propose to model the metric tensor with the inverse multiquadratics kernel function $\mathcal{K}$ used by (Tolstikhin et al., 2018) and a Gaussian kernel, that is,

$$\mathcal{K}_m(\mathbf{z}, \mathbf{c_k}) = C/(C + ||\mathbf{z} - \mathbf{c_k}||_2^2)$$
$$\mathbf{k} = \text{argmin}\, ||\mathbf{z} - \mathbf{c_k}||_2^2$$
$$\mathcal{K}_g(\mathbf{z}, \mathbf{c_k}) = \exp\left(-\beta_k ||\mathbf{z} - \mathbf{z_k}||_2^2\right)$$

where $\mathbf{c_k}, k = 1, 2, 3, ..., K$ are clusters of latent codes, and β_k is the bandwidth. We observe that the inverse multiquadratics kernel $\mathcal{K}_m$ generally performs better. We use the above kernels as constraints over the Jacobian-determinant, so that,

$$|det \frac{\partial f'}{\partial \mathbf{z}}| = |1 + \mathbf{u^T}(\mathbf{z})\phi(\mathbf{z})\mathbf{w}| \cdot \mathcal{K}(\mathbf{z}, \mathbf{c_k}) \quad (16)$$

As we explained earlier in this section, latent representation of region outside of input manifold should have high curvature in latent space. During training, we seek to maximize this regularized Jacobian $|det \frac{\partial f'}{\partial \mathbf{z}}|$ rather than the original one. This ensures that those latent codes within latent clusters, and therefore very likely to be on or near input manifold in input space, have much smaller curvature $|det\ G| = |\frac{\partial f}{\partial \mathbf{z}}|^2$ than those outside of latent clusters, as those outside of manifold would seek larger Jacobian in order to counter-effect the regularization term $\mathcal{K}$. The latent space $\mathcal{Z}'$ transformed by normalizing flow f is thus curved with respect to input manifold. This type of normalizing flow thus learns a latent space to respect geometric characteristics of input space. The KL vanishing problem is then unlikely to happen with a curved latent space. This is because most high-dimensional data in real life forms a curved manifold which is unlikely sampled from a multivariate standard Gaussian. Then, if the latent space reflects curvature of a curved manifold, the support of latent codes certainly does not follow a standard Gaussian either. This helps to push the posterior $q(\mathbf{z}|\mathbf{x})$ away from the standard Gaussian and never collapse to a non-informative prior.

4.2 RNF Wasserstein Autoencoder

Here we consider using the Wasserstein objective to model the latent marginal distribution of a curved latent space learned by an RNF. The Wasserstein objective with MMD is appealing in our case for two main reasons.

First, instead of minimizing the KL-divergence $KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$, it minimizes distance between $q(\mathbf{z}) = \int q(\mathbf{z}|\mathbf{x})p(\mathbf{x})dx$ and $p(\mathbf{z})$, which encourages the marginal distribution of latent space to be as close as the prior while not affecting individual posterior distribution $q(\mathbf{z}|\mathbf{x})$ conditioned on each input. This makes the KL-divergence between posterior and prior, and equivalently the mutual information between latent codes and input sentences $I(\mathbf{z}, \mathbf{x}) = KL(q(\mathbf{z}, \mathbf{x})||q(\mathbf{z})p(\mathbf{x})) = \mathbb{E}_{p(\mathbf{x})}[KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))] - KL(q(\mathbf{z})||p(\mathbf{z}))$ impossible to be vanished as the objective does not require it to be small. Since the learned latent codes and input sentences have non-zero mutual information, the generative model will not ignore latent codes when generating texts.

Second, the MMD regularization in WAE makes it possible to optimize normalizing flow without computing the Jacobian-determinant explicitly. The use of MMD is necessary as getting a closed form KL divergence is no-longer possible after we apply RNF to the posterior. And, since the generative function G in the reconstruction term of Wasserstein objective can be any function or composition of functions (Tolstikhin et al., 2018), we can easily compose an RNF function into G such that the reconstructed texts are $\bar{X} = G(f(Z)) = G(Z'), Z' \in \mathcal{Z}'$.

Now, given a series of RNF $F = f_K \circ ... \circ f_1$, and let $\mathcal{Z}_K$ be the **curved** latent space after applying K flows over the original latent space $\mathcal{Z}$, we optimize the following RNF-Wasserstein objective,

$$D(P_X, P_G) = \inf_{Q(Z|X) \in \mathcal{Q}} \mathbb{E}_{P_X}\, \mathbb{E}_{Q(Z|X)}[c(X, G(Z'))] + \lambda MMD(Q_{Z'}, P_{Z'}) + \alpha(KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) - \sum \log |det \frac{\partial f'}{\partial \mathbf{z}}|) \quad (17)$$

where $Z \sim \mathcal{Z}$, $Z' \sim \mathcal{Z}'$, and $Z' = F(Z)$. We approximate MMD term with the Gaussian kernel $k(\mathbf{z}, \mathbf{z}') = e^{-||\mathbf{z}-\mathbf{z}'||^2}$, that is, $MMD(p, q) = \mathbb{E}_{p(\mathbf{z}),p(\mathbf{z}')}[k(\mathbf{z}, \mathbf{z}')] + \mathbb{E}_{q(\mathbf{z}),q(\mathbf{z}')}[k(\mathbf{z}, \mathbf{z}')] - 2\,\mathbb{E}_{p(\mathbf{z}),q(\mathbf{z}')}[k(\mathbf{z}, \mathbf{z}')]$.

Here we choose to minimize the MMD distance

Model	PTB		YAHOO		YELP	
	NLL(KL)	PPL	NLL(KL)	PPL	NLL(KL)	PPL
LSTM-LM **	116.2 (-)	104.2	334.9 (-)	66.2	- -	
VAE	105.2 (1.74)	121.1	339.2 (0.01)	69.9	198.6 (0.01)	55.0
VAE-NF	96.8 (0.87)	82.9	353.8 (0.10)	83.0	200.4 (0.10)	62.5
lagging-VAE **	- -		**326.6 (6.70)**	**64.9**	–	
vmf-VAE	96.0 (5.70)	79.6	359.3 (17.9)	89.9	198.0 (6.40)	54.0
WAE-RNF	**91.9 (15.4)**	**66.1**	**339.0 (3.00)**	**71.6**	**183.9 (12.7)**	**41.1**

Table 1: Language Modeling results on PTB, YAHOO and YELP 13 Reviews. ** are results gathered from (Yang et al., 2017; Xiao et al., 2018; He et al., 2019). Negative log-likelihood (NLL) is approximated by its lower bound, where the number in parentheses indicates KL-divergence. NF stands for the standard planar normalizing flow without Riemannian curvature.

Data	PTB				Yelp			
Model	**NLL**	**Re-KL**	**Log-J**	**PPL**	**NLL**	**Re-KL**	**Log-J**	**PPL**
WAE	104.9	–(1.9)	–	131.	198.5	–(1.9)	–	55
WAE-NF	92.3	14.3	14.3	67.3	184.3	13.9	14.2	41.4
WAW-RNF	**91.9**	**15.4**	**15.2**	**66.1**	**183.9**	**12.7**	**12.1**	**41.1**

Table 2: Language Modeling using WAE-RNF. We report NLL, PPL, Sum of Log Jacobian, and KL divergence between $q(\mathbf{z}'|\mathbf{x})$ and $p(\mathbf{z}')$.

between the prior P_Z and the marginal of non-curved latent space. This makes sampling procedure for generation tasks much easier, as it is easy to sample a latent code from a non-informative prior. We can get a sample $\mathbf{z}'$ from $\mathcal{Z}'$ indirectly by sampling: $\mathbf{z} \sim P_Z(\mathbf{z})$ and $\mathbf{z}' \sim F(\mathbf{z})$. On the other hand, it would be much more difficult to sample from a curved latent space $\mathcal{Z}'$ directly as the only prior knowledge we have about $\mathcal{Z}'$ is the curvature reflected by RNF implicitly and hence we do not know the support of $Q(Z')$.

5 Experimental Results

In this section, we investigate WAE's performance with Riemannian Normalizing Flow over language and text modeling.

5.1 Datasets

We use Penn Treebank (Marcus et al., 1993), Yelp 13 reviews (Xu et al., 2016), as in (Xu and Durrett, 2018; Bowman et al., 2015), and Yahoo Answers used in (Xu and Durrett, 2018; Yang et al., 2017) to follow and compare with prior studies. We limit the maximum length of a sample from all datasets to 200 words. The datasets statistics is shown in Table 3.

Data	Train	Dev	Test	Vocab
PTB	42068	3370	3761	10K
Yelp13	62522	7773	8671	15K
Yahoo	100K	10K	10K	20K

Table 3: Datasets statistics; The numbers reflect size of each dataset. Vocab is the vocabulary size.

5.2 Experimental Setup

For each model, we set the maximum vocabulary size to 20K and the maximum length of input to 200 across all data sets. Following Bowman et al. (2015), we use one-layer undirectional LSTM for both encoder-decoder models with hidden size 200. Latent codes dimension is set to 32 for all models. We share Word Embeddings of size 200. For stochastic encoders, both MLP_μ and MLP_σ are two layer fully-connected networks with hidden size 200 and a batch normalizing output layer (Ioffe and Szegedy, 2015).

We use Adam (Kingma and Ba, 2015) with learning rate set to 10^{-3} to train all models. Dropout is used and is set to 0.2. We train all models for 48 epochs, each of which consists of 2K steps. For models other than WAE, KL-annealing is applied and is scheduled from 0 to 1 at the 21st epoch.

For vmf-VAE (Xu and Durrett, 2018), we set the word embedding dimension to be 512 and the hidden units to 1024 for Yahoo, and set both of them to 200 for PTB and Yelp. The temperature κ is set to 80 and is kept constant during training.

For all WAE models, we add a small KL divergence term to control the posterior distribution. We found that if we only use RNF with MMD as the distance metric, then the posterior may diverge from the prior such that no reasonable samples can be generated from a standard Gaussian variable.

Hence, for all data sets, we schedule the KL divergence weight α from 0 to 0.8, and the weight of the MMD term is set as $\lambda = 10 - \alpha$. β_k of RBF is set to 10 for all models. For RNF, we use pretrained standard VAE models to gather the clusters $\mathbf{c_k}$, $k = 1, ..., K$, of latent codes, where we set the number of clusters to be 20. We use three normalizing flow for all experiments.

Hyperparameter of $\mathbf{K}_m$ When using the inverse multiquadratics kernel $\mathcal{K}_m(\mathbf{z}, \mathbf{c}_k) = C/(C + ||\mathbf{z} - \mathbf{c}_k||_2^2)$ for RNF, we follow the choice of hyperparameter in (Tolstikhin et al., 2018). We set $C = 2 \cdot d \cdot s$, where d is the dimensionality of latent codes $\mathbf{z}$, and s is ranged in $(0.1, 0.2, 0.5, 1, 2, 5, 10)$. The final kernel is computed by $\mathcal{K}_m(\mathbf{z}, \mathbf{c}_k) = \sum_s 2ds/(2ds + ||\mathbf{z} - \mathbf{c}_k||_2^2)$. As explained by (Tolstikhin et al., 2018), this strategy allows us to explore a wider range of hyperparameter in one setting.

5.3 Language Modeling Results

We show the language modeling results for PTB, Yahoo and Yelp in Table 1. We compare negative log-likelihood (NLL), KL divergence, and perplexity (PPL) with all other existing methods. The negative log-likelihood is approximated by its lower bound.

We use the negative of ELBO to approximate NLL for all VAE models. For those with normalizing flows, we use the modified ELBO, which is $\mathcal{L} = \mathbb{E}_{q(\mathbf{z}^{(0)}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}^{(T)}) - \log q(\mathbf{z}^{(0)}|\mathbf{x}) + \log p(\mathbf{z}^{(T)})] + \mathbb{E}_{q(\mathbf{z}^{(T)})}[\sum_{t=1}^{T} log|\frac{\partial f^{(t)}}{\partial \mathbf{z}^{(t-1)}}|]$.

The numbers show that KL-annealing and dropout used by Bowman et al. (2015) are helpful for PTB, but for complex datasets such as Yahoo and Yelp, the KL divergence still drops to zero due to the over-expressiveness of LSTM. This phenomenon is not alleviated by applying normalizing flow to make the posterior more flexible, as shown in the third row. Part of the reason may be that a simple NF such as a planar flow is not flexible enough and is still dominated by a powerful LSTM decoder.

We find that the KL vanishing is alleviated a little bit if using WAE, which should be the case as WAE objective does not require small KL. We also find that simply applying a planar flow over WAE does not improve the performance that much. On the other hand, using RNF to train WAE dramatically helps the situation which achieves the lowest text perplexity on most conditions except for

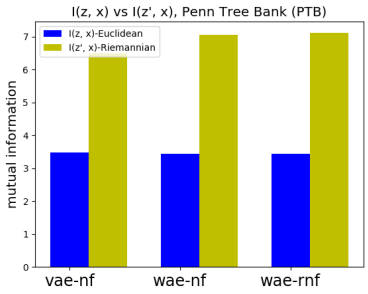

Figure 3: PTB. Comparison between the amount of mutual information stored in latent codes for different models.

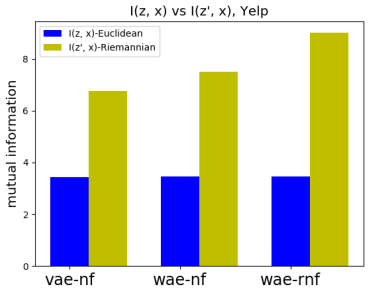

Figure 4: Yelp. Comparison between the amount of mutual information stored in latent codes for different models.

YAHOO Answers, where (He et al., 2019; Yang et al., 2017; Kim et al., 2018) have the current state-of-the-art results. We want to emphasize that CNN-VAE (Yang et al., 2017) and SA-VAE (Kim et al., 2018) are not directly comparable with other current approaches. Here, we compare with models that use LSTM as encoder-decoder and have similar time complexity, while the use of CNN as decoder in CNN-VAE would dramatically change the model expressiveness, and it is known that SA-VAE's time complexity (Kim et al., 2018; He et al., 2019) is much higher than all other existing approaches.

5.4 How Good is Riemannian Latent Representation?

Mutual information between $\mathcal{Z}'$ and $\mathcal{X}$ One important question is how useful are latent variables. Since no metrics are perfect (Wang et al., 2018), we should not just look at sample perplexity to judge how good a latent code is. Hence, we also investigate how much information can be encoded into latent codes. We believe that the mutual information term $I(\mathbf{z}; \mathbf{x})$ is a better metric regarding the usefulness of latent codes than sample

the company said it will be sold to the company 's promotional programs and _UNK
the company also said it will sell $ n million of soap eggs turning millions of dollars
the company said it will be _UNK by the company 's _UNK division n
the company said it would n't comment on the suit and its reorganization plan
this is a reflection of socialism and capitalism
the company also said it will sell its _UNK division of the company 's _UNK
earlier this year the company said it will sell $ n billion of assets and _UNK to the u.s
last year he said the company 's earnings were n't disclosed

one of my favorite places to eat at the biltmore . the food is good . and the food is good.
very good food . the food was very good . the service was great and the food is very good.
one of my favorite places to eat and a great breakfast spot . the food is great . the staff is friendly.
took a few friends to join me to the _UNK . i was n't sure what to expect.
one of my favorite places to eat at the biltmore . the food is good , the service was great .
i love the fact that they have a lot took a few friends to join me to the _UNK .
i have been to this location a few times , but i ' ve never been disappointed
let me start by saying that i love the idea of how to describe it .

Table 4: Qualitative comparison between VAE and our proposed approach. First row: PTB samples generated from prior $p(\mathbf{z})$ by VAE (*upper half*) and WAE-RNF (*lower half*). Second row: Yelp samples generated from prior $p(\mathbf{z})$ by VAE (*upper half*) and WAE-RNF (*lower half*).

perplexity, as it tells us directly how much information we can infer from $\mathbf{x}$ by looking $\mathbf{z}$.

We use Monte Carlo method (Metropolis and Ulam, 1949) to get an approximation of $I(\mathbf{z}, \mathbf{x}) = KL(q(\mathbf{z}, \mathbf{x})||q(\mathbf{z})p(\mathbf{x})) = \mathbb{E}_{p(\mathbf{x})}[KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))] - KL(q(\mathbf{z})||p(\mathbf{z}))$. We compared mutual information between input $\mathbf{x}$ and latent codes $\mathbf{z}$ sampled from Euclidean latent space $\mathcal{Z}$ and Riemannian latent space $\mathcal{Z}'$ respectively. We see that even though NF does not necessarily help WAE to achieve the lowest perplexity, it does make latent codes to preserve more information about the input. For WAE trained with RNF, sample perplexity and mutual information metric are both good. It is cleary that $I(\mathbf{z}', \mathbf{x}) > I(\mathbf{z}, \mathbf{x})$, where $\mathbf{z}'$ is sampled from the curved space, and $\mathbf{z}$ is the sample transformed by the normal planar flow. This further strengthens our confidence over the usefulness of the curved latent space $\mathcal{Z}'$.

Generating Texts from latent spaces Another way to explore latent space is to look at the quality of generated texts. Here we compare sentences generated from methods that do not use Wasserstein objective and RNF with those generated from curved latent space $\mathcal{Z}'$ learned by WAE.

We observe that texts generated from flat Euclidean space are not as diverse as the ones generated from curved space learned by WAE-RNF. This is largely related to the nature of Wasserstein objective. In WAE, the KL-divergence $KL(q||p))$ between the prior and the posterior $q(\mathbf{z}|\mathbf{x})$ conditioned on each input $\mathbf{x}$ does not need to be small to optimize the Wasserstein objective. This indicates that the marginal $q(\mathbf{z})$ is able to match to the prior $p(\mathbf{z})$ while allowing each posterior $q(\mathbf{z}|\mathbf{x})$ to have a much more diverse support than that of a standard Gaussian $p(\mathbf{z})$. Therefore, if we randomly generate samples from curved latent space $\mathcal{Z}'$ many times, we are likely to get samples scattered in different support of distinct posterior conditioned on different input $\mathbf{x}$. Hence, the reconstructed sentences will have a much more diverse meaning or structure.

6 Conclusion

In this paper, we introduced Riemannian Normalizing Flow to train Wasserstein Autoencoder for text modeling. This new model encourages learned latent representation of texts to respect geometric characteristics of input sentences space. Our results show that RNF WAE does significantly improve the language modeling results by modeling the Riemannian geometric space via normalizing flow.

Acknowledgement

We want to thank College of Creative Studies and Gene & Lucas Undergraduate Research Fund for providing scholarships and research opportunities for Prince Zizhuang Wang. We also want to thank Yanxin Feng from Wuhan University for helpful discussion about Riemannian Geometry, and Yunxian He (CMU), Wenhu Chen (UCSB), and Yijun Xiao (UCSB) for their comments which helped us improve our paper and experiments.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. 2018. Sylvester normalizing flows for variational inference. *UAI*.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. 2018. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*.

Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel, and Bernhard Schoelkopf. 2017. From optimal transport to generative modeling: the vegan cookbook. *arXiv preprint arXiv:1705.07642*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *CoNLL*.

Shuyang Gao, Rob Brekelmans, Greg Ver Steeg, and Aram Galstyan. 2018. Auto-encoding total correlation explanation. *arXiv preprint arXiv:1802.05822*.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. *ICML*.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *JMLR*.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *ICLR*.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. 2018. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. iclr 2015. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Nicholas Metropolis and Stanislaw Ulam. 1949. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICML*.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *ICML*.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. *EMNLP*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. 2018. Wasserstein auto-encoders. *ICLR*.

Jakub M Tomczak and Max Welling. 2016. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*.

Cédric Villani. 2008. *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *ACL*.

Yijun Xiao, Tiancheng Zhao, and William Yang Wang. 2018. Dirichlet variational autoencoder for text modeling. *arXiv preprint arXiv:1811.00135*.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuangjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. *EMNLP*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. *EMNLP*.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2image: Conditional image generation from visual attributes. In *ECCV*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2017a. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *ACL*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017b. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*.