CS294-158 Deep Unsupervised Learning
UC Berkeley, Spring 2019

HW3: Variational Autoencoders

Due: March 14, 11:59pm

1 VAEs in 2D

1.1 Part A

In this problem, you will investigate when a VAE places information into its latent code. Run this snippet generate two datasets:

```
import numpy as np
def sample_data_1():
    count = 100000
    rand = np.random.RandomState(0)
    return [[1.0, 2.0]] + rand.randn(count, 2) * [[5.0, 1.0]]
def sample_data_2():
    count = 100000
    rand = np.random.RandomState(0)
    return [[1.0, 2.0]] + (rand.randn(count, 2) * [[5.0, 1.0]]).dot(
        [[np.sqrt(2) / 2, np.sqrt(2) / 2], [-np.sqrt(2) / 2, np.sqrt(2) / 2]])
```

Train one VAE in this configuration on both datasets:

- 2D latent variables $\mathbf{z}$ with a standard normal prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$.
- An approximate posterior $q_\theta(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu_\theta(\mathbf{x}), \Sigma_\theta(\mathbf{x}))$, where $\mu_\theta(\mathbf{x})$ is a mean vector and $\Sigma_\theta(\mathbf{x})$ and is a diagonal covariance matrix
- A decoder $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$, where $\mu_\theta(\mathbf{z})$ is a mean vector and $\Sigma_\theta(\mathbf{z})$ and is a diagonal covariance matrix

and train another in this configuration, also on both datasets:

- Prior and approximate posterior identical to above
- A decoder $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu_\theta(\mathbf{z}), \sigma_\theta^2(\mathbf{z})I)$, where $\mu_\theta(\mathbf{z})$ is a mean vector and $\sigma_\theta^2(\mathbf{z})$ is a *scalar*.

Deliverables for all 4 choices of datasets and models:

1. Loss curves and final loss, in bits per dimension. Include separate curves and numbers for the full ELBO, the KL term $\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}[\mathrm{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))]$, the decoder term $\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}[-\log p(\mathbf{x}|\mathbf{z})]$.
2. Samples from the full generation path $(\mathbf{z}\sim p(\mathbf{z}), \mathbf{x}\sim p(\mathbf{x}|\mathbf{z}))$ and without decoder noise $(\mathbf{z}\sim p(\mathbf{z}), \mathbf{x}=\mu(\mathbf{z}))$. Draw both in the same plot.
3. Is the VAE using the latent code? How do you know? If it does, what about the data does it capture qualitatively?

1.2 Part B

Run the following code to generate data for training. It will generate a dataset of labeled samples: the first element of the returned tuple is the data in $\mathbb{R}^2$, and the second element of the returned tuple is the label in $\{0, 1, 2\}$.

You will train VAEs on unlabeled samples, and you will use the labels for visualization purposes only. Take the first 80% of the samples as a training set and the remaining 20% as a test set.

```
import numpy as np
def sample_data_3():
    count = 100000
    rand = np.random.RandomState(0)
    a = [[-1.5, 2.5]] + rand.randn(count // 3, 2) * 0.2
    b = [[1.5, 2.5]] + rand.randn(count // 3, 2) * 0.2
    c = np.c_[2 * np.cos(np.linspace(0, np.pi, count // 3)),
              -np.sin(np.linspace(0, np.pi, count // 3))]
    c += rand.randn(*c.shape) * 0.2
    data_x = np.concatenate([a, b, c], axis=0)
    data_y = np.array([0] * len(a) + [1] * len(b) + [2] * len(c))
    perm = rand.permutation(len(data_x))
    return data_x[perm], data_y[perm]
```

Train a VAE with a 2-dimensional $\mathbf{z}\sim p(\mathbf{z}) = N(\mathbf{z}; 0, I)$ and a Gaussian encoder and decoder (with diagonal covariance matrices).

Provide these deliverables:

1. Loss curves and final loss, in bits per dimension. Include separate curves and numbers for the full ELBO, the KL term, and the decoder term; do so on the train and validation sets. What is the final test set performance (all three numbers)?
2. Display samples.
3. Display latents for data. To do so, take *labeled* training data $(\mathbf{x}, y)$ and plot $\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x})$, colored by the label y. Comment on the appearance of the latent spaces for the two models.
4. Pick the first 100 points in the test set. Evaluate and report the IWAE objective using 100 samples, and compare to the standard ELBO (i.e. IWAE with 1 sample).

2 High-dimensional data

In this question, you will be training the Variational Auto-Encoder on the SVHN dataset. The SVHN dataset is split into train, valid and test samples: 65931, 7326 and 26032 respectively.

Here's the recommended architecture (all convolutions are implemented with 'same' padding) :

```
Residual_stack():
    for _ in range(5):
        relu(),
        conv2d(n_filters=64, kernel_size=(3, 3), strides=(1, 1)),
        relu(),
        conv2d(n_filters=64*2, kernel_size=(3, 3), strides=(1, 1)),
        gated_shortcut_connection() # https://arxiv.org/pdf/1612.08083.pdf,
    relu()

Encoder():
    conv2d(n_filters=128, kernel_size=(4, 4), strides=(2, 2)),
    relu(),
    conv2d(n_filters=256, kernel_size=(4, 4), strides=(2, 2)),
    relu(),
    conv2d(n_filters=256, kernel_size=(3, 3), strides=(1, 1)),
    Residual_stack()

Decoder()
    conv2d(n_filters=256, kernel_size=(3, 3), strides=(1, 1))
    Residual_stack(),
    conv2d_transpose(n_filters=128, kernel_size=(4, 4), strides=(2, 2)),
    relu(),
    conv2d_transpose(n_filters=3, kernel_size=(4, 4), strides=(2, 2))
```

a. Train a VAE with a diagonal covariance Gaussian decoder. The architecture above outputs one value per color channel. You will have to change it to learn both the mean and variance of each pixel. Choose the prior to be a standard normal gaussian and the approximate posterior to be a diagonal covariance Gaussian. b. With the same encoder and decoder as in a., modify the prior to be an autoregressive flow. This is precisely explained in [1] (Equations 12, 13 and 14). For reference, here is an implementation that's stable: Model the autoregressive flow on the prior as a 2D PixelCNN with 8x8 spatial dimension. A shallow PixelCNN with type A 3x3 mask in the first layer followed by 3 residual layers of type B mask (as in Homework 1) is used to construct the scale and translate terms of the flow per spatial dimension. Using an Adam Optimizer with a learning rate of 2e-4 and applying tanh nonlinearities to logstds are recommended practices.

For **both** these models, report the following (and point out if the autoregressive prior makes a difference and if it does, why):

1. Variational Lower Bound on train, validation and test (units: bits/dim) over the course of training.

2. 100 samples from the trained models.
3. Interpolations between 5 pairs of test samples from the test set.

3 Bonus

Implement a limited receptive field autoregressive PixelCNN decoder with discretized mixture of logistics output distribution. You will find [1] highly relevant.

References

[1] Xi Chen et al. "Variational lossy autoencoder". In: *arXiv preprint arXiv:1611.02731* (2016).