Deep learning

7.4. Variational autoencoders

François Fleuret

https://fleuret.org/dlc/

Coming back to generating a signal, instead of training an autoencoder and modeling the distribution of Z, we can try an alternative approach:

Impose a distribution for Z and then train a decoder g so that $g(Z)$ matches the training data.

Notes

We saw in lecture 7.2. "Deep Autoencoders" that autoencoders can, to some extent, model the data distribution by first mapping the data to a smaller dimension latent space, and then fitting a density model.
As seen with the experiments on MNIST, this is not satisfying because,

- either the latent space is of very small dimension, in which case it makes sense to use a parameterized density model over the latent representation, but the input signal is not modeled properly, or
- the latent space is of higher dimension, in which case it becomes harder to properly model the latent representation.

We consider two distributions:

- p is the distribution on $\mathcal{X} \times \mathbb{R}^d$ of a pair (X, Z) composed of an encoding state $Z \sim \mathcal{N}(0, I)$ and the output of the decoder g on it.

- q is the distribution on $\mathcal{X} \times \mathbb{R}^d$ of a pair (X, Z) composed of a sample X taken from the data distribution and the output of the encoder on it,

Our goal is that $p(X)$ mimics the data-distribution $q(X)$, that is to find g that maximizes the log-likelihood

$$\frac{1}{N} \sum_n \log p(x_n) = \hat{\mathbb{E}}_{q(X)} \Big[\log p(X) \Big].$$

However, while we can sample z and compute $g(z)$ for complicated gs, **we cannot compute $p(x)$ for a given x**, and even less compute its derivatives.

Notes

In what follows, we will use notations such as $q(X)$ for marginals, and $q(Z \mid X = x)$ for conditionals.
The quantity $\hat{\mathbb{E}}_{q(X)}$ here is the empirical estimation on the training data, since the said training data are i.i.d $\sim q(X)$.

The **Variational Autoencoder** proposed by Kingma and Welling (2013) relies on a tractable approximation of this log-likelihood.

Note that their framework involves **stochastic** encoder f, and decoder g, whose outputs depend on both their inputs and additional randomness.

Remember that $q(X)$ is the data distribution, and $f(x) \sim q(Z \mid X = x)$.

We want to maximize

$$\mathbb{E}_{q(X)}\Big[\log p(X)\Big],$$

and it can be shown that

$$\log p(X = x) \geq \underbrace{\mathbb{E}_{q(Z|X=x)}\Big[\log p(X = x \mid Z)\Big] - \mathbb{D}_{\mathrm{KL}}(q(Z \mid X = x) \,\|\, p(Z))}_{\text{“Evidence lower bound” (ELBO)}}.$$

So it makes sense to maximize

$$\mathbb{E}_{q(X,Z)}\Big[\log p(X \mid Z)\Big] - \mathbb{E}_{q(X)}\Big[\mathbb{D}_{\mathrm{KL}}(q(Z \mid X) \,\|\, p(Z))\Big].$$

Notes

We can derive the ELBO using the KL-divergence, which is always positive

$$\mathbb{D}_{\mathrm{KL}}(a\|b) = -\mathbb{E}_{a(u)}\left[\log \frac{b(u)}{a(u)}\right] = -\int_u a(u) \log \frac{b(u)}{a(u)} du \geq -\int_u a(u)\left(\frac{b(u)}{a(u)} - 1\right) du = 0.$$

$$\begin{aligned}
&\log p(X = x)\\
&= \mathbb{E}_{q(z|X=x)}\left[\log p(X = x)\right]\\
&= \mathbb{E}_{q(z|X=x)}\left[\log p(X = x)\frac{p(z \mid X = x)}{p(z \mid X = x)}\frac{q(z \mid X = x)}{q(z \mid X = x)}\right]\\
&= \mathbb{E}_{q(z|X=x)}\left[\log \frac{p(z)p(X = x \mid z)}{p(z \mid X = x)}\frac{q(z \mid X = x)}{q(z \mid X = x)}\right]\\
&= \mathbb{E}_{q(z|X=x)}\left[\log p(X = x \mid z) + \log \frac{p(z)}{q(z \mid X = x)} + \log \frac{q(z \mid X = x)}{p(z \mid X = x)}\right]\\
&= \mathbb{E}_{q(z|X=x)}\left[\log p(X = x \mid z)\right] + \mathbb{E}_{q(z|X=x)}\left[\log \frac{p(z)}{q(z \mid X = x)}\right] - \mathbb{E}_{q(z|X=x)}\left[\log \frac{p(z \mid X = x)}{q(z \mid X = x)}\right]\\
&= \mathbb{E}_{q(z|X=x)}\left[\log p(X = x \mid z)\right] - \mathbb{D}_{\mathrm{KL}}\left(q(z \mid X = x)\|p(z)\right) + \underbrace{\mathbb{D}_{\mathrm{KL}}\left(q(z \mid X = x)\|p(z \mid X = x)\right)}_{\geq 0}\\
&\geq \mathbb{E}_{q(z|X=x)}\left[\log p(X = x \mid z)\right] - \mathbb{D}_{\mathrm{KL}}\left(q(z \mid X = x)\|p(z)\right)
\end{aligned}$$

So the final loss is

$$\mathscr{L} = \mathbb{E}_{q(X)}\Big[\mathbb{D}_{\mathsf{KL}}(q(Z \mid X) \,\|\, p(Z))\Big] - \mathbb{E}_{q(X,Z)}\Big[\log p(X \mid Z)\Big],$$

with

- $q(X)$ is the data distribution
- $p(Z) = \mathcal{N}(0, I)$.

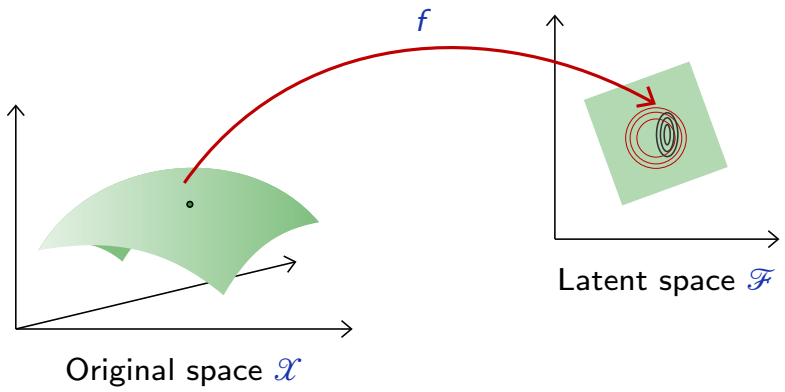
Kingma and Welling propose that both the encoder f and decoder g map to a Gaussian with diagonal covariance. Hence they map to twice the dimension (e.g. $f(x) = (\mu^f(x), \sigma^f(x))$) and

- $q(Z \mid X = x) \sim \mathcal{N}(\mu^f(x), \mathrm{diag}(\sigma^f(x)))$
- $p(X \mid Z = z) \sim \mathcal{N}(\mu^g(z), \mathrm{diag}(\sigma^g(z)))$.

The first term of $\mathscr{L}$ is the average of

$$\mathbb{D}_{\mathrm{KL}}\big(\underbrace{q(Z \mid X = x)}_{\mathcal{N}(\mu^f(x),\sigma^f(x))} \,\|\, \underbrace{p(Z)}_{\mathcal{N}(0,I)}\big) = -\frac{1}{2}\sum_d \Big(1 + 2\log\sigma_d^f(x) - \big(\mu_d^f(x)\big)^2 - \big(\sigma_d^f(x)\big)^2\Big).$$

over the x_ns.



The first term of $\mathscr{L}$ is the average of

$$\mathbb{D}_{\mathrm{KL}}\big(\underbrace{q(Z \mid X = x)}_{\mathcal{N}(\mu^f(x),\sigma^f(x))} \,\|\, \underbrace{p(Z)}_{\mathcal{N}(0,I)}\big) = -\frac{1}{2}\sum_d \Big(1 + 2\log\sigma_d^f(x) - \big(\mu_d^f(x)\big)^2 - \big(\sigma_d^f(x)\big)^2\Big).$$

over the x_ns.

This can be implemented as

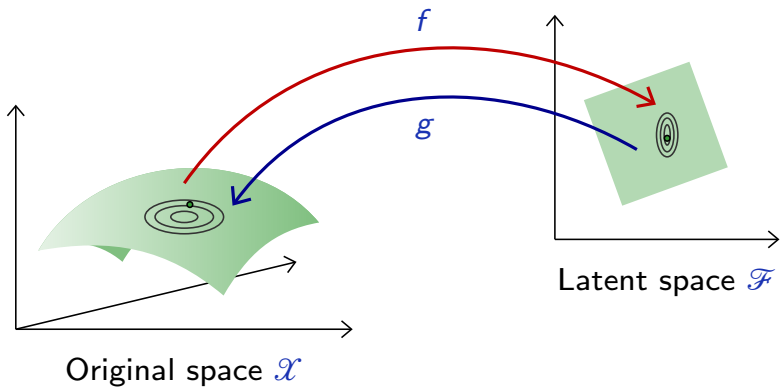
```
param_f = model.encode(input)
mu_f, logvar_f = param_f.split(param_f.size(1)//2, 1)

kl = - 0.5 * (1 + logvar_f - mu_f.pow(2) - logvar_f.exp())
kl_loss = kl.sum() / input.size(0)
```

As Kingma and Welling (2013), we use a constant variance of 1 for the decoder, so the second term of $\mathscr{L}$ becomes the average of

$$-\log p(X = x \mid Z = z) = \frac{1}{2}\sum_d (x_d - \mu_d^g(z))^2 + \text{cst}$$

over the x_n, with one z_n sampled for each, i.e.

$$z_n \sim \mathcal{N}\left(\mu^f(x_n), \sigma^f(x_n)\right), \quad n = 1, \ldots, N.$$



As Kingma and Welling (2013), we use a constant variance of 1 for the decoder, so the second term of $\mathscr{L}$ becomes the average of

$$-\log p(X = x \mid Z = z) = \frac{1}{2}\sum_{d}(x_d - \mu_d^g(z))^2 + \text{cst}$$

over the x_n, with one z_n sampled for each, i.e.

$$z_n \sim \mathcal{N}\left(\mu^f(x_n), \sigma^f(x_n)\right), \quad n = 1, \ldots, N.$$

This can be implemented as

```
std_f = torch.exp(0.5 * logvar_f)
z = torch.randn_like(mu_f) * std_f + mu_f
output = model.decode(z)

fit = 0.5 * (output - input).pow(2)
fit_loss = fit.sum() / input.size(0)
```

We had for the standard autoencoder

```
z = model.encode(input)
output = model.decode(z)
loss = 0.5 * (output - input).pow(2).sum() / input.size(0)
```

and putting everything together we get for the VAE

```
param_f = model.encode(input)
mu_f, logvar_f = param_f.split(param_f.size(1)//2, 1)

kl = - 0.5 * (1 + logvar_f - mu_f.pow(2) - logvar_f.exp())
kl_loss = kl.sum() / input.size(0)

std_f = torch.exp(0.5 * logvar_f)
z = torch.randn_like(mu_f) * std_f + mu_f
output = model.decode(z)

fit = 0.5 * (output - input).pow(2)
fit_loss = fit.sum() / input.size(0)

loss = kl_loss + fit_loss
```

During inference we do not sample, and instead use μ^f and μ^g as prediction.

Notes

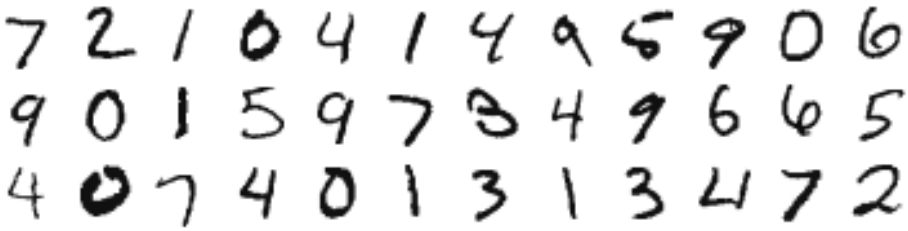
The `kl_loss` aims at making the distribution in the embedding space close to the normal density. The fitting loss `fit_loss` aims at making the reconstructed data point correct in a probabilistic sense: the original data point should be likely under the Gaussian that we get when we come back from the latent space.

Note in particular the **re-parameterization trick**:

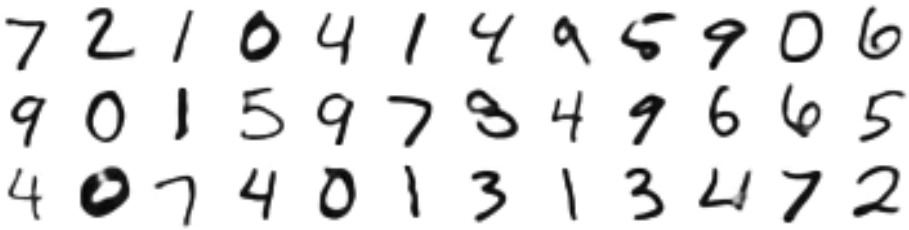
```
z = torch.randn_like(mu_f) * std_f + mu_f
output = model.decode(z)
```

Implementing the sampling of z that way allows to compute the gradient w.r.t f's parameters without any particular property of `normal_()`.

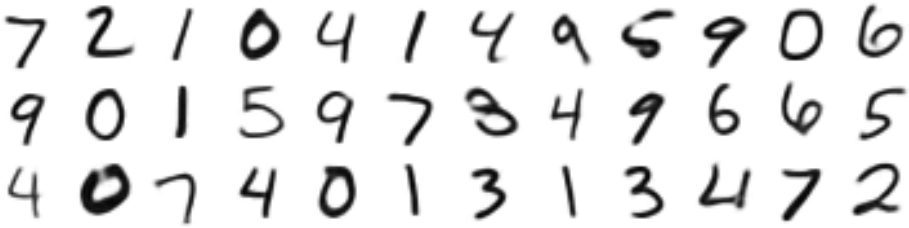
Original



Autoencoder reconstruction ($d = 32$)



Variational Autoencoder reconstruction ($d = 32$)

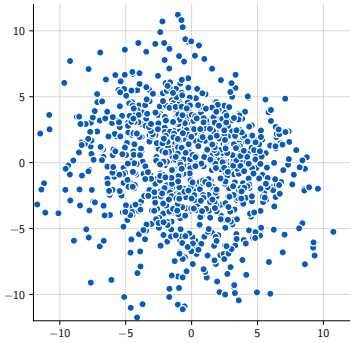


Notes

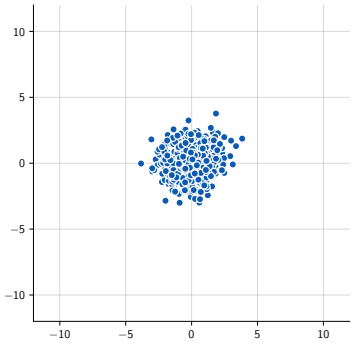
The images at the top are original test MNIST samples. The images in the middle are the reconstructed samples with the standard autoencoder as seen in lecture 7.2. "Deep Autoencoders". The images at the bottom are the reconstructed samples obtained with the variational autoencoder.
The results are not as good as with the standard autoencoder, which is not surprising since there is an additional constraint on the distribution in the latent space.

We can look at two latent features to check that they are Normal for the VAE.

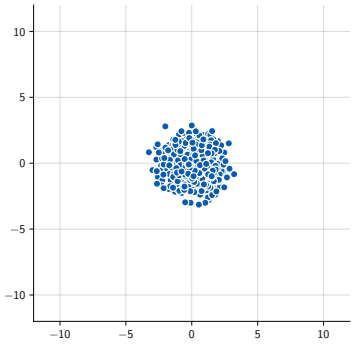
Autoencoder



Variational autoencoder



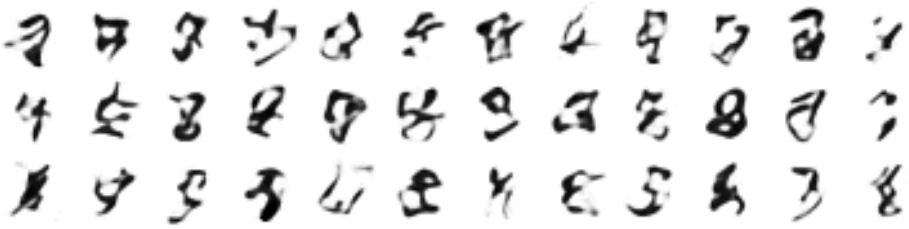
$\mathcal{N}(0,1)$



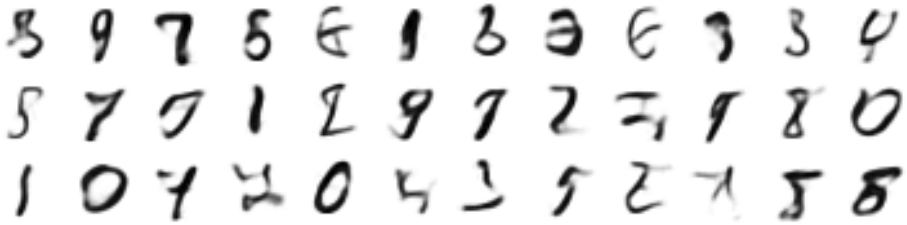
Notes

The first scatter plot on the top-left shows the empirical distribution of two latent dimensions when the encoder is from the standard autoencoder. The plot was generated by choosing at random two dimensions among the 32 ones of the latent space, encode 1000 MNIST samples, and draw a point at the two resulting coordinates. We can see that this is not a normal distribution. If the same process is done with the variational autoencoder we get the scatter plot at the bottom left, that we can compare with a similar scatter plot obtained by sampling independent normal coordinates, shown at the bottom right.
The variational autoencoder did its job of making the embedded representation follow a Gaussian distribution.

Autoencoder sampling ($d = 32$)



Variational Autoencoder sampling ($d = 32$)



Notes

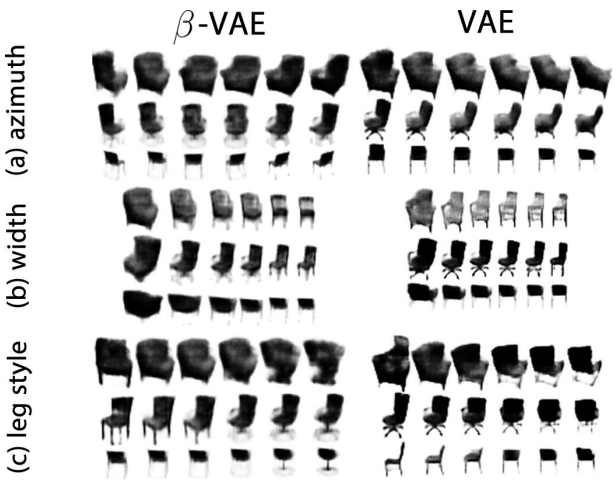
We can compare the result of sampling data points in the latent space and map them back in the original space with the decoder. To that purpose, we first generate a random Gaussian vector of dimension equal to that of the latent space, $d = 32$, and then we run the decoder on that random sample, which produces an image in the original space.
The images on the top are obtained with the decoder from the standard autoencoder, and the images on the bottom are for the variational autoencoder. Although these small-scale experiments are far from the state of the art, the latter samples are more realistic.

Making the embedding $\sim \mathcal{N}(0, 1)$, often results in "disentangled" representations.

This effect can be reinforced with a greater weight of the KL term

$$\mathscr{L} = \beta \, \mathbb{E}_{q(X)}\Big[\mathbb{D}_{\mathsf{KL}}(q(Z \mid X) \,\|\, p(Z))\Big] - \mathbb{E}_{q(X,Z)}\Big[\log p(X \mid Z)\Big],$$

resulting in the β-VAE proposed by Higgins et al. (2017).



(Higgins et al., 2017)

(Higgins et al., 2017)

References

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. **beta-vae: Learning basic visual concepts with a constrained variational framework**. In International Conference on Learning Representations (ICLR), 2017.

D. P. Kingma and M. Welling. **Auto-encoding variational bayes**. CoRR, abs/1312.6114, 2013.