(Variational) Autoencoder

Representation learning

Need good representations for effective learning

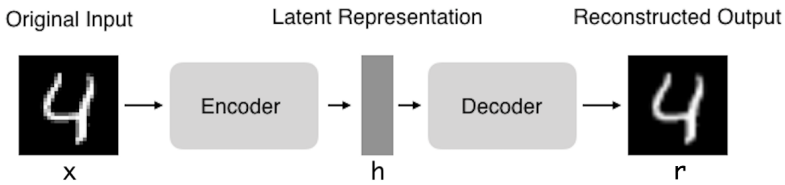
E.g. Making features independent, dimensionality reduction

Preserve as much information as possible, while attaining nice properties

Learn good representation from unlabeled data, then solve the supervised problem

Autoencoder

Goal: reconstruct the input



Encoder:

- Maps the input to the latent space
- $h = f(x)$

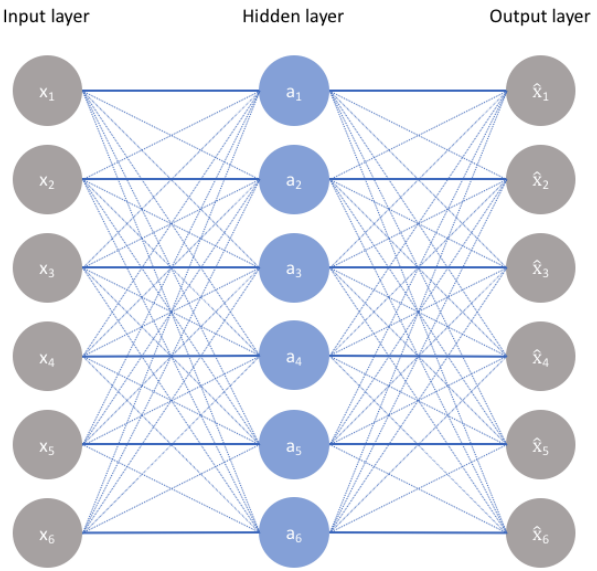
Decoder:

- Reconstruct the input from the latent space representation
- $r = g(h)$

We want $\hat{x} = g(f(x))$ to be close to x.

Constraints

We hope we can discover dependencies in the data.



We need to constrain the model to prevent memorizing.

Autoencoder

An ideal autoencoder is

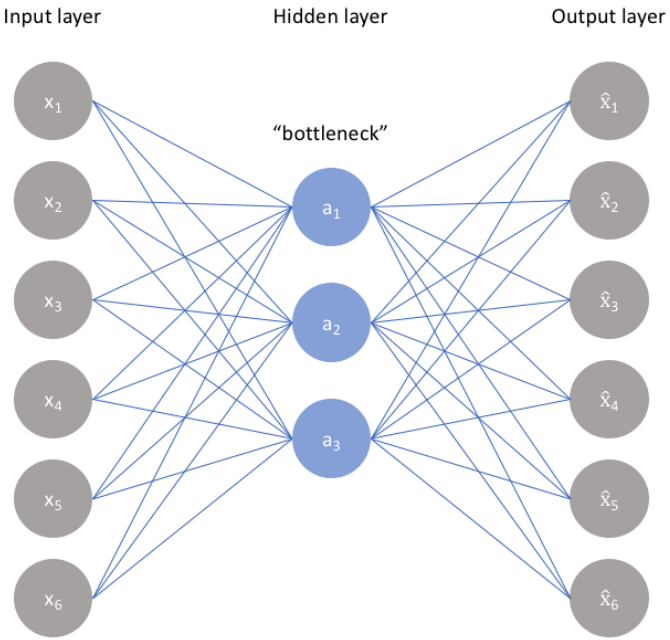
- Sensitive enough to the inputs to acccurately build a reconstruction
- Insensitive enough so the model doesn't simply memorize the data.

In many cases the loss function consists of two parts:

- Reconstruction loss $\mathcal{L}(x, g(f(x)))$
- Regularizer, which discourages memorizing.

Undercomplete autoencoder

Easiest way is to constrain the number of nodes in the hidden layers.

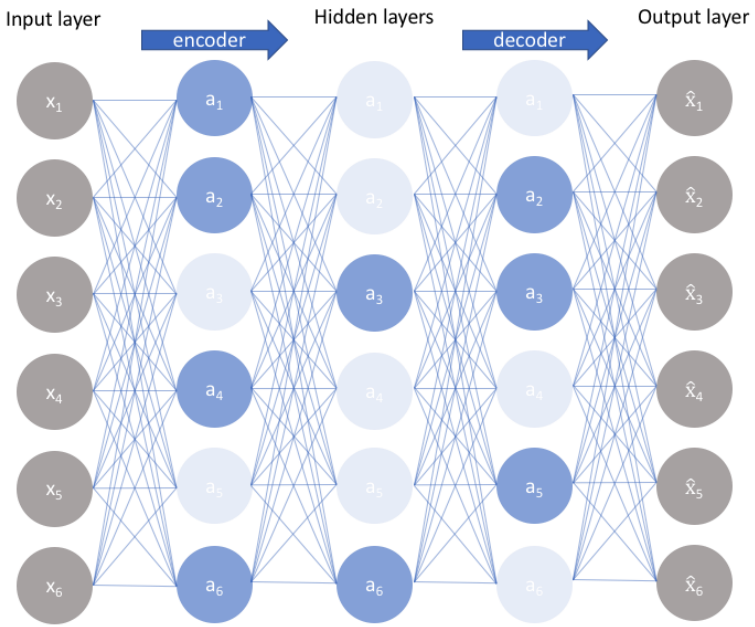


Limits the amount of information that can flow through the network.

No regularizer, only reconstruction loss.

Sparse autoencoder

Regularized autoencoder: we have constraints, but not on the dimensions.



Only a few nodes are activated.

Sparse autoencoder

Loss function: Reconstruction loss + sparsity constraint

Two types:

- L_1 regularization

sparsity constraint = $\lambda \sum_i |a_i^{(h)}|$

λ is a hyperprameter, $a_i^{(h)}$ is the activation of the ith node in the hidden layer.

- Kullback-Leibler divergence

Kullback-Leibler divergence

Measures distance between two distributions

$$D_{\mathrm{KL}}(p\|q) = \int p(x)\cdot\log\frac{p(x)}{q(x)}\,dx = \int p(x)\cdot\big(\log p(x) - \log q(x)\big)\,dx$$

$$D_{\mathrm{KL}}(p\|q) = E_{x\sim p}\log\frac{p(x)}{q(x)}$$

- Not symmetric
- Triangle inequality doesn't hold
- nonegative

Sparse autoencoder

ρ: sparsity parameter: small number

For all nodes in the hidden layer

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} (a_j^{(h)}(x^{(i)}))$$

Sparsity constraint: $\sum_{j=1}^{s_h} D_{\mathrm{KL}}(\rho \| \hat{\rho}_j)$, Kullback-Leibler divergence berween two Bernoulli distributions

$$D_{\mathrm{KL}}(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}$$

Loss function:

$$\mathcal{L}(x, \hat{x}) + \beta \sum_{j=1}^{s_h} D_{\mathrm{KL}}(\rho \| \hat{\rho}_j),$$

Compression

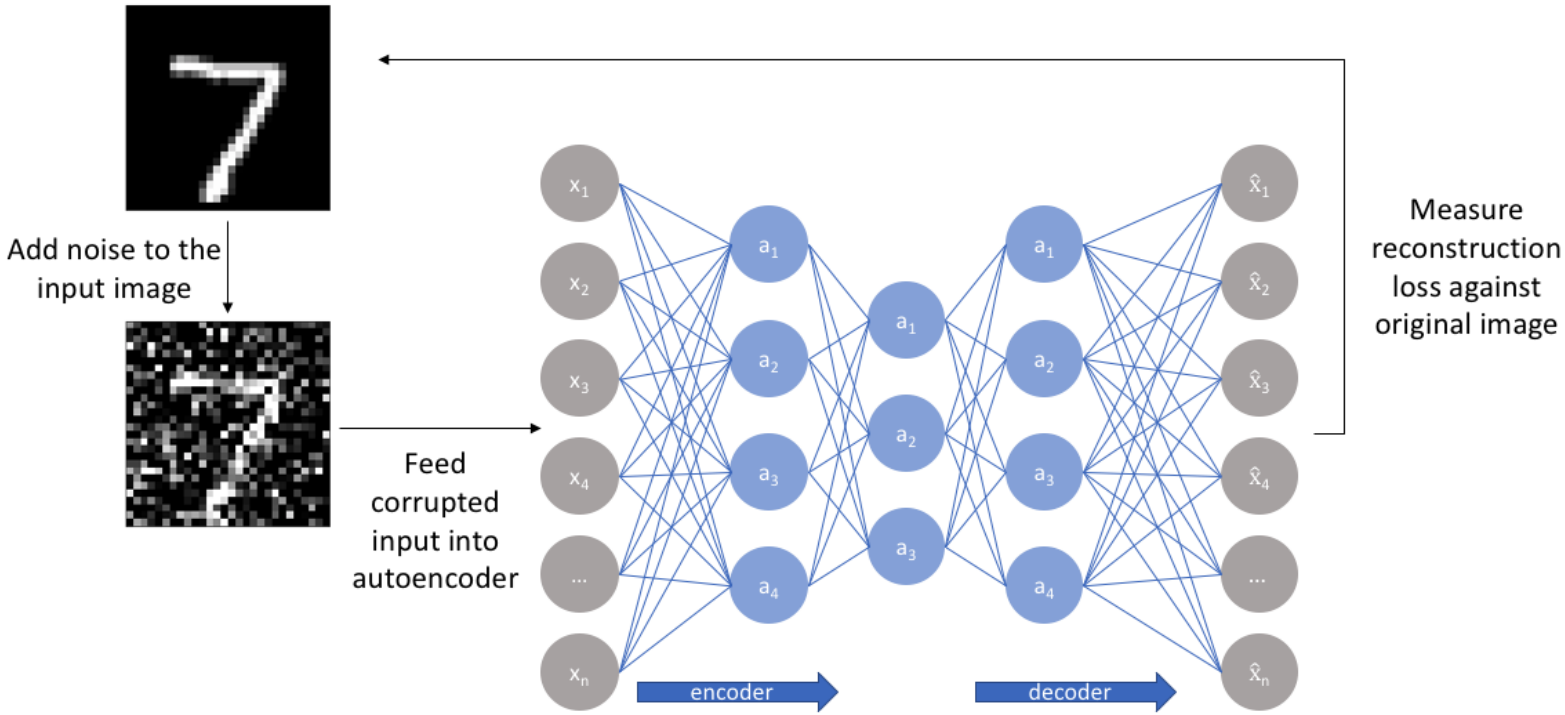
Compression and decompression functions are

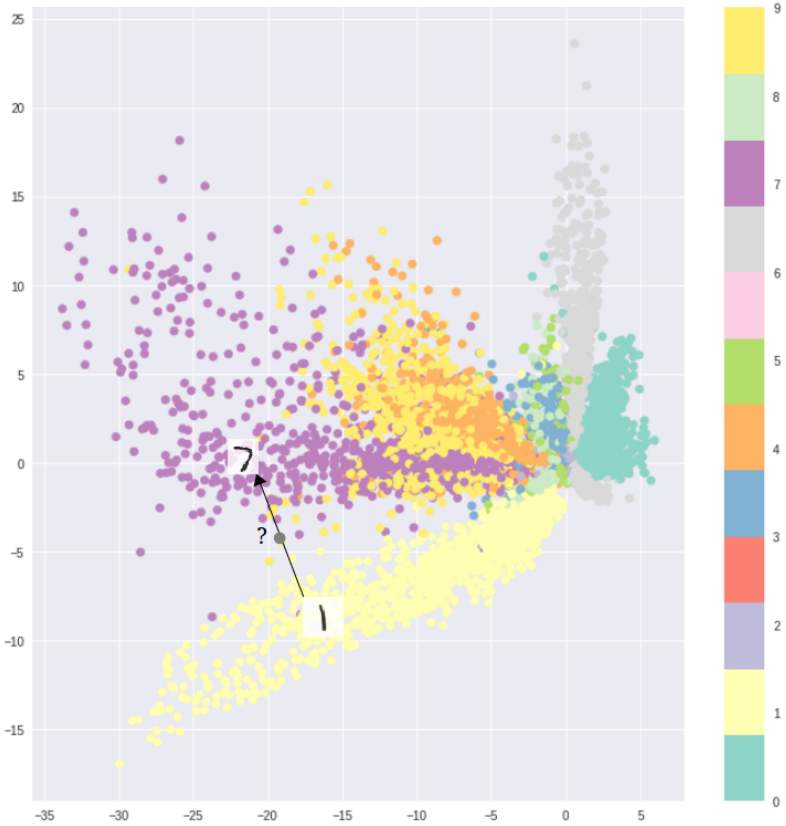
- data-specific
- lossy
- learned automatically from examples

Not good for data compression:

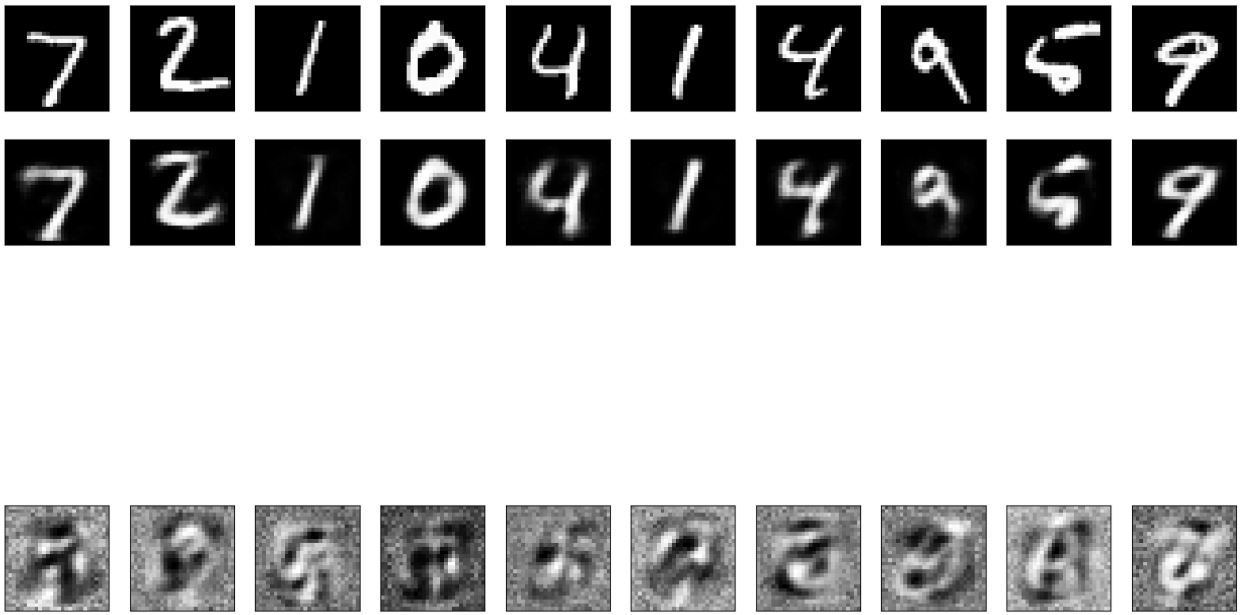
- data-specific
- Very hard to train one, which does a better job than JPEG

Denoising autoencoder





Training an autoencoder on the MNIST dataset, and visualizing the encodings from a 2D latent space reveals the formation of distinct clusters. This makes sense, as distinct encodings for each image type makes it far easier for the decoder to decode them.



Variational autoencoder

Generative model:

Goal: approximate the underlying $p^*(x)$ distribution of the training set.

Want to generate new samples, similar to the training set.

Latent variable model:

Learn a low-dimensional latent representation

We assume it generated the actual training data.

Problem scenario

We have a dataset $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{n}$ consisting of n i.i.d samples from some continuous or discrete variable x.

We assume the data was generated by a random process, involving an unobserved continuous variable $\mathbf{z}$.

1. A value $\mathbf{z}^{(i)}$ is generated from some prior distribution $p(\mathbf{z})$
2. A value $\mathbf{x}^{(i)}$ is generated from some conditional distribution $p(\mathbf{x} \mid \mathbf{z})$.

$$p(x) = \int p(x|z)p(z)\ dz$$

We aim to maximize the probability of each $x^{(i)}$ in the training set.

We are interested in a general algorithm, which works in the case of

1. Intractability: the marginal likelihood:

$$p(x) = \int p(x \mid z)p(z)dz$$

and the posterior density:

$$p(z \mid x) = \frac{p(x, z)}{p(x)}$$

are both intractable,

2. A large dataset: we have so much data that sampling-based solutions would be too slow.

$$p(x) \approx \frac{1}{N} \sum_{i=1}^{N} p(x|z_i)$$

The prior and the likelihood

We assume that samples of z can be drawn from a simple distribution, i.e $\mathcal{N}(0, I)$.

Also, we assume that the likelihood is from a parametric family with parameters θ.

$$p_\theta(x|z) = \mathcal{N}(x|f(z,\theta), \sigma^2 \cdot I),$$

so it is Gaussian with mean $f(z,\theta)$ and covariance $\sigma^2 \cdot I$.

We want to optimize the θ parameters, so the logarithm of the marginal likelihood will be maximal:

$$\log p_\theta(x^{(1)}, \ldots, x^{(n)}) = \sum_{i=1}^{n} \log p_\theta(x^{(i)}).$$

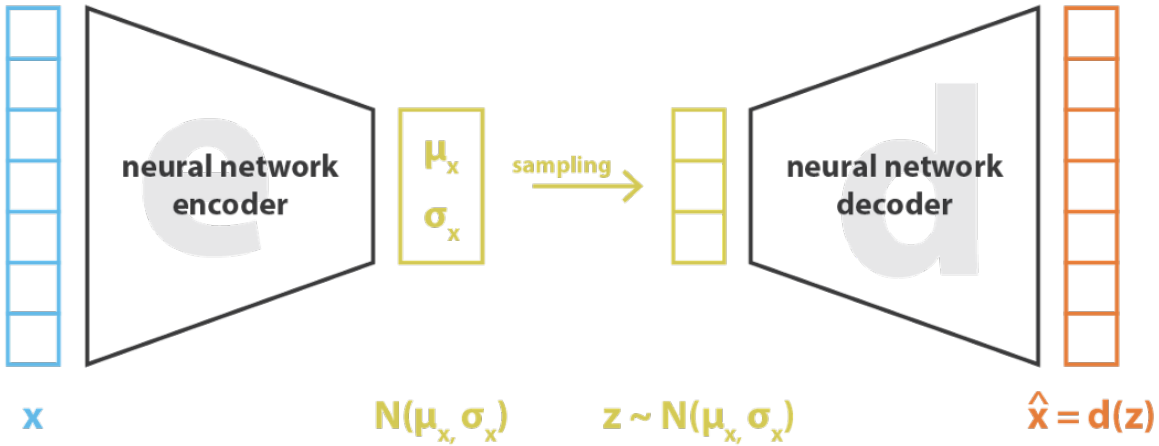
$$p(x^{(i)}) \approx \frac{1}{\ell} \sum_j p(x^{(i)}|z_j)$$

In practice, for most z, the probability $p_\theta(x^{(i)} \mid z)$ will be almost zero, so they contribute nothing to $p_\theta(x^{(i)})$.

The key idea behind the variational autoencoder is that we would like to sample values of z that are likely to have produced $x^{(i)}$.

So we want to infer the characteristics of z, but we can only see the $x^{(i)}$.

We cannot use the posterior $p_\theta(z \mid x^{(i)})$ because is intractable, so we will approximate it with a new distribution, $q_\phi(z \mid x^{(i)})$.



$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

We compute the Kullback-Leibler divergence between $p_\theta(z \mid x^{(i)})$ and $q_\phi(z|x^{(i)})$:

$$D_{KL}(q_\phi(z|x^{(i)})\|p_\theta(z|x^{(i)})) = E_{q_\phi(z|x^{(i)})}(\log q_\phi(z|x^{(i)}) - \log p_\theta(z|x^{(i)})) =$$

Applying Bayes rule to $p_\theta(z \mid x^{(i)})$:

$$= E_{q_\phi(z|x^{(i)})}(\log q_\phi(z|x^{(i)}) - \log p_\theta(x^{(i)}|z) - \log p(z)) + \log p_\theta(x^{(i)})$$

After rearranging:

$$\log p_\theta(x^{(i)}) - D_{KL}(q_\phi(z|x^{(i)})\|p_\theta(z|x^{(i)})) =$$

$$= E_{q_\phi(z|x^{(i)})}\big(\log p_\theta(x^{(i)}|z) + \log p(z) - \log q_\phi(z|x^{(i)})\big) =$$

Variational lower bound

$$= E_{q_\phi(z|x^{(i)})} \log p_\theta(x^{(i)}|z) - D_{KL}(q_\phi(z|x^{(i)})\|p(z)).$$

We call the right hand side the variational lower bound:

$$\mathcal{L}(\theta, \phi, x^{(i)}) = E_{q_\phi(z|x^{(i)})} \log p_\theta(x^{(i)}|z) - D_{KL}(q_\phi(z|x^{(i)})\|p(z)).$$

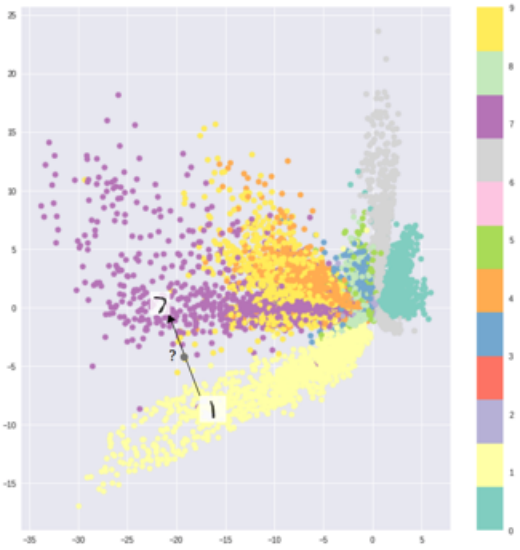
This is indeed a lower bound for $\log p_\theta(x^{(i)})$, because the KL-divergence is always nonnegative.

We want to maximize $\log p_\theta(x^{(i)})$, so we need to optimize the lower bound with respect to both parameters, θ and ϕ.

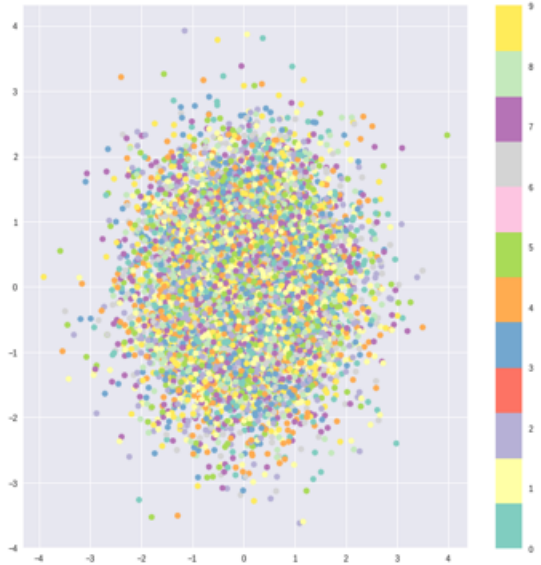
We want to perform stochastic gradient descent on the right hand side.

Latent space

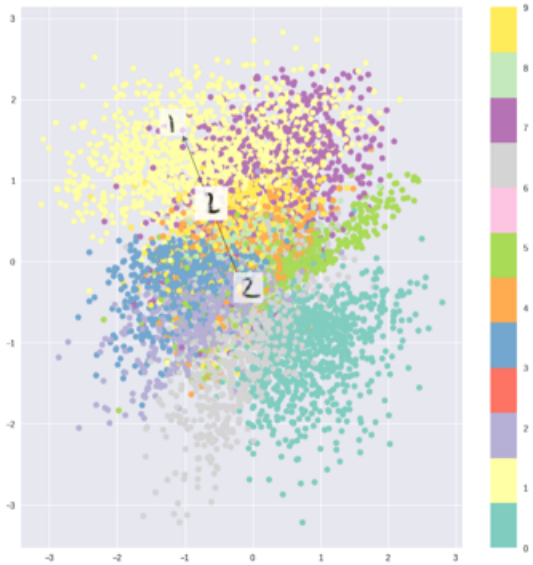
Only reconstruction loss



Only KL divergence



Combination



The usual choice for $q_\phi(z|x^{(i)})$ is $\mathcal{N}(z \mid \mu_\phi(x^{(i)}), \Sigma_\phi(x^{(i)})$, where μ_ϕ and Σ_ϕ are arbitrary, deterministic functions that can be learned from data.

Now the KL-divergence on the right hand side is between two multivariate Gaussian distributions, which can be computed in closed form:

$$D_{KL}(\mathcal{N}(\mu_0, \Sigma_0) \| \mathcal{N}(\mu_1, \Sigma_1)) =$$

$$= \frac{1}{2}\left(\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1}(\mu_1 - \mu_0) - k + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right),$$

where k is the dimensionality of the distribution.

In our case:

$$D_{KL}(\mathcal{N}(\mu(x^{(i)}), \Sigma(x^{(i)}))\|\mathcal{N}(0, I)) =$$

$$= \frac{1}{2}\Big(\mathrm{tr}(\Sigma(x^{(i)})) + (\mu(x^{(i)})^T(\mu(x^{(i)}) - k + \log\det(\Sigma(x^{(i)}))$$

We could use sampling to estimate the other term on the right hand side, but instead as is standard in stochastic gradient descent we take one sample of z and treat $p_\theta(x^{(i)} \mid z)$ for that z as an approximation of $E_{q_\phi(z|x^{(i)})} \log p_\theta(x^{(i)} \mid z)$.

So we compute the gradient of

$$\log p_\theta(x^{(i)} \mid z) - D_{KL}(q_\phi(z \mid x^{(i)})\|p(z)),$$

and average it over arbitrarily many samples of z.

Reparametrization trick

The problem with he last equation is that $E_{q_\phi(z|x^{(i)})} \log p_\theta(x^{(i)} \mid z)$ depends on the parameters ϕ also, but here the dependency has disappeared.

We need to backpropagate the error through a layer that samples z from $q_\phi(z \mid x)$, which is a non-continuous operation and has no gradient.

Solution: we move the sampling to an input layer.

Given $\mu(x^{(i)})$ and $\Sigma(x^{(i)})$, tha mean and covariance of $q_\phi(z \mid x^{(i)})$, we can sample from $\mathcal{N}(\mu(x^{(i)}), \Sigma(x^{(i)}))$ by first sampling $\varepsilon \sim \mathcal{N}(0, I)$, and then computing $z = \mu(x^{(i)}) + \Sigma^{1/2}(x^{(i)}) \cdot \varepsilon$.

So the equation becomes:

$$E_{\varepsilon\sim\mathcal{N}(0,I)}\Big(\log p_\theta(x^{(i)} \mid z = \mu(x^{(i)}) + \Sigma^{1/2}(x^{(i)}) \cdot \varepsilon) -$$

$$-D_{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z))\Big)$$