String Theory meets Machine Learning - (Variational) Autoencoder

Robin Schneider

Uppsala University

November 2020

Unsupervised learning

What is unsupervised learning?

- Cluster analysis, anomaly detection, learning latent variables, generating data
- Supervised learning is conditional probability $p_\theta(y|X)$, unsupervised is learning true distribution $p_\theta(X) \approx p^*(X)$
- Self learning

Unsupervised learning

What is unsupervised learning?

- Cluster analysis, anomaly detection, learning latent variables, generating data
- Supervised learning is conditional probability $p_\theta(y|X)$, unsupervised is learning true distribution $p_\theta(X) \approx p^*(X)$
- Self learning

Algorithms

- K-means clustering
- Principal component analysis
- Generative Adversial Networks (GANs)
- (Variational) Autoencoder
- (Persistent homology)

Autoencoders

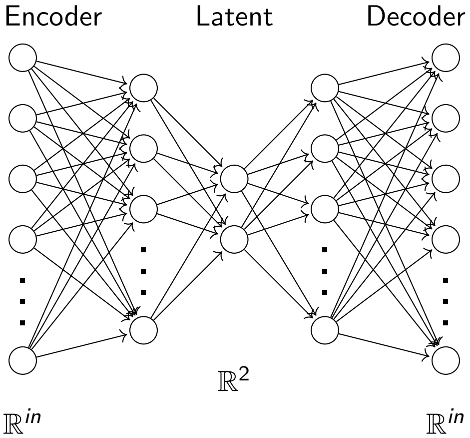

Figure: *A fully connected autoencoder with two latent dimensions.*

Applications

What are autoencoders good for?

- Data denoising
- Data compression to latent dimension
- Data visualization (clustering)
- Anomaly detection

Let's go back to out favourite data set MNIST.

MNIST autoencoder

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 64)        640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 2)                 2050
=================================================================
Total params: 76,546
Trainable params: 76,546
Non-trainable params: 0
```

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 3136)              9408
_________________________________________________________________
reshape (Reshape)            (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 14, 14, 64)        36928
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 28, 28, 64)        36928
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 28, 28, 1)         577
=================================================================
Total params: 83,841
Trainable params: 83,841
Non-trainable params: 0
```

Figure: *Neural network architecture for encoder and decoder applied to MNIST data set with two latent dimensions.*

MNIST denoising

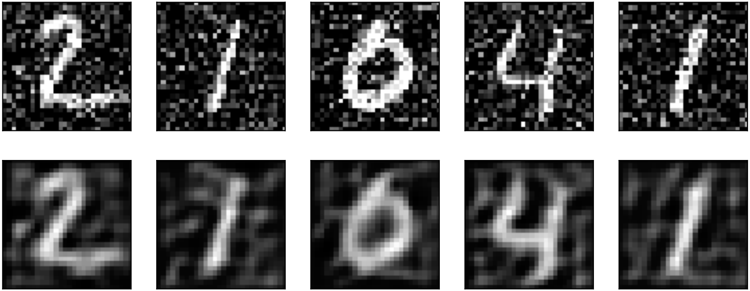

Figure: *On the top row noisy hand written numbers of the MNIST data set. On the bottom row the denoised pictures after running through an auto encoder.*

MNIST compression

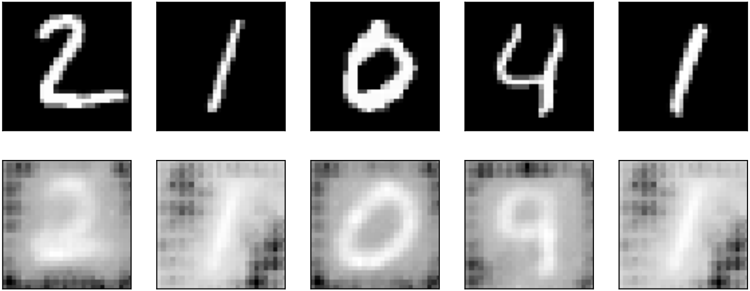

Figure: *(De-)compression of MNIST images to two latent dimensions. On the top row five hand written digits and on the bottom after running through the encoder.*

MNIST visualization

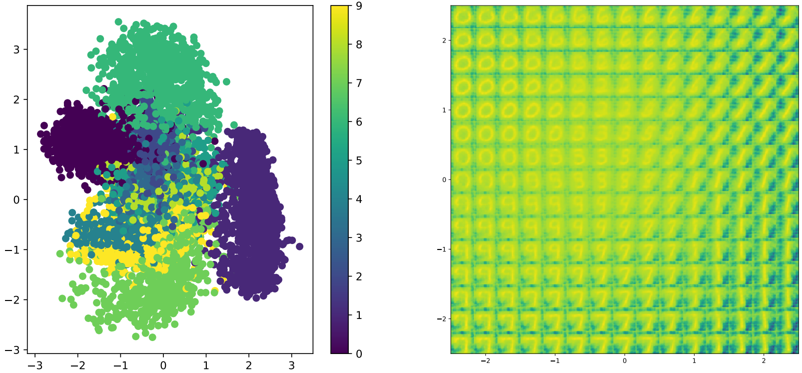

Figure: *On the left clustering of MNIST digits to two latent dimensions via encoder, on the right decoded images of samples from the two latent dimensions.*

Standard Like Model

Heterotic string compactification with three ingredients [1106.4804,1202.1757,1307.4787].

1. Calabi Yau manifold M.
2. Line bundle sum $V = \otimes_{a=1}^{5} L_a$.
3. Freely acting discrete symmetry Γ for Wilson line.

For example:

$$\mathcal{M}_{5302} = \left[\begin{array}{c||ccc} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{array}\right]^{6,30}_{-48} \quad \text{and } V = \begin{bmatrix} -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & -2 \\ 1 & 1 & -2 & 1 & -1 \\ 1 & 1 & 0 & -2 & 0 \end{bmatrix}$$

and $|\Gamma| = 4$. There are a total of 17329 such models.

SLM autoencoder

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 330)	0
dense (Dense)	(None, 32)	10592
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 8)	136
dense_3 (Dense)	(None, 4)	36
dense_4 (Dense)	(None, 2)	10

Total params: 11,302
Trainable params: 11,302
Non-trainable params: 0

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 4)	12
dense_6 (Dense)	(None, 8)	40
dense_7 (Dense)	(None, 16)	144
dense_8 (Dense)	(None, 32)	544
dense_9 (Dense)	(None, 330)	10890
reshape (Reshape)	(None, 30, 11)	0

Total params: 11,630
Trainable params: 11,630
Non-trainable params: 0

Figure: *Neural network architecture for encoder and decoder applied to SLM data set with two latent dimensions.*

SLM visualization

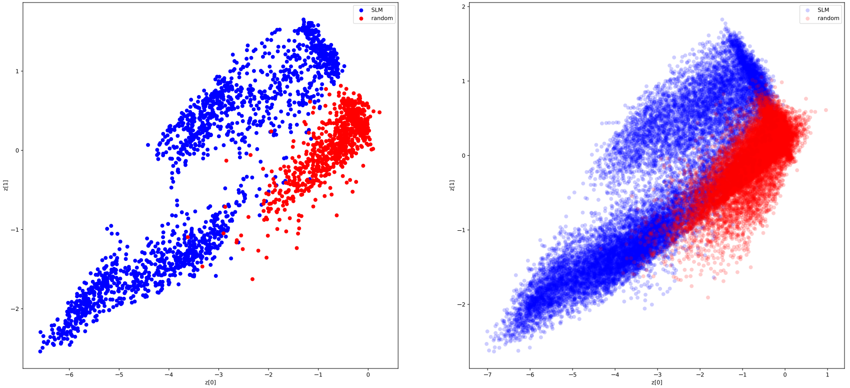

Figure: *Image of clustered Standard like models. First investigated in [2003.13339]. To the left with norm $V < 5$. To the right all SLMs.*

Variational Autoencoders

Variational Autoencoders are generative modeling $p_\theta(X|z)$. They

- combine stochastic gradient descent and Bayesian inference into deep generative models [1312.6114,1401.4082,1906.02691],
- are with GANs the most popular generative model,
- are used in Physics, e.g. at CERN or in astronomy.

Variational Autoencoders

Variational Autoencoders are generative modeling $p_\theta(X|z)$. They

- combine stochastic gradient descent and Bayesian inference into deep generative models [1312.6114,1401.4082,1906.02691],
- are with GANs the most popular generative model,
- are used in Physics, e.g. at CERN or in astronomy.

Starting point: Assume there are some latent variables z controlling your data X. Take a deterministic function (decoder) $f(\theta) : z \to X$.
Here z is a random variable. Then $f(z, \theta)$ also becomes random.

Probability Density Function

Assume that $p(z)$ and $p_\theta(X|z)$ follow some PDF we can then marginalize

$$p_\theta(X) = \int p_\theta(X, z)dz = \int p_\theta(X|z)p(z)dz \tag{1}$$

This integral is often intractable, see e.g [1601.00670].

Probability Density Function

Assume that $p(z)$ and $p_\theta(X|z)$ follow some PDF we can then marginalize

$$p_\theta(X) = \int p_\theta(X, z)dz = \int p_\theta(X|z)p(z)dz \tag{1}$$

This integral is often intractable, see e.g [1601.00670].
We take

$$p_\theta(X|z) = \mathcal{N}(X|\mu = f(z, \theta), \sigma^2) \tag{2}$$

and introduce an approximation to posterior $p_\theta(z|X)$, an inference model

$$q_\phi(z|x) = \mathcal{N}(X|\mu, \sigma^2), \text{ with } (\mu, \sigma^2) \sim h(X, \phi) \tag{3}$$

Assume there exists no single good interpretation of z, hence the prior is

$$p(z) = \mathcal{N}(0, I). \tag{4}$$

ELBO and KL

Want to maximize $\log p_\theta(X)$:

$$
\begin{aligned}
\mathbb{E}_{q_\phi(z|X)}[\log p_\theta(X)] &= \mathbb{E}_{q_\phi(z|X)}\left[\log \frac{p_\theta(X,z)}{p_\theta(z|X)}\right] \\
&= \mathbb{E}_{q_\phi(z|X)}\left[\log \frac{p_\theta(X,z)}{q_\phi(z|X)}\frac{q_\phi(z|X)}{p_\theta(z|X)}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(z|X)}\left[\log \frac{p_\theta(X,z)}{q_\phi(z|X)}\right]}_{:=\mathcal{L}_{\theta,\phi}(x)} + \underbrace{\mathbb{E}_{q_\phi(z|X)}\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]}_{:=D_{KL}(q_\phi(z|X)||p_\theta(z|X))}
\end{aligned}
\tag{5}
$$

[1] Recall entropy: $H = -\sum_{i=1}^{N} p(x_i) \cdot \log p(x_i)$

ELBO and KL

Want to maximize $\log p_\theta(X)$:

$$\begin{aligned}
\mathbb{E}_{q_\phi(z|X)}[\log p_\theta(X)] &= \mathbb{E}_{q_\phi(z|X)}\left[\log \frac{p_\theta(X,z)}{p_\theta(z|X)}\right] \\
&= \mathbb{E}_{q_\phi(z|X)}\left[\log \frac{p_\theta(X,z)}{q_\phi(z|X)} \frac{q_\phi(z|X)}{p_\theta(z|X)}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(z|X)}\left[\log \frac{p_\theta(X,z)}{q_\phi(z|X)}\right]}_{:=\mathcal{L}_{\theta,\phi}(x)} + \underbrace{\mathbb{E}_{q_\phi(z|X)}\left[\log \frac{q_\phi(z|X)}{p_\theta(z|X)}\right]}_{:=D_{KL}(q_\phi(z|X)||p_\theta(z|X))}
\end{aligned} \tag{5}$$

First term is Evidence Lower BOund (ELBO), second term is Kullback-Leibler divergence, which measures the relative 'difference' between two probability distributions (always positive)[1]. Want to **maximize** ELBO 1. maximizes log likelihood, 2. minimize KL divergence.

[1] Recall entropy: $H = -\sum_{i=1}^{N} p(x_i) \cdot \log p(x_i)$

Optimizing and Monte Carlo sampling

Gradient descent on ELBO

$$\nabla_\theta \mathcal{L}_{\theta,\phi} = \nabla_\theta \mathbb{E}_{q_\phi(z|X)}\left[\log \frac{p_\theta(X,z)}{q_\phi(z|X)}\right] \simeq \nabla_\theta \log p_\theta(X,z) \tag{6}$$

works just fine. However, $\nabla_\phi \mathcal{L}_{\theta,\phi}$ more tricky, since we can't pull ∇_ϕ into $\mathbb{E}_{q_\phi(z|X)}$.

Optimizing and Monte Carlo sampling

Gradient descent on ELBO

$$\nabla_\theta \mathcal{L}_{\theta,\phi} = \nabla_\theta \mathbb{E}_{q_\phi(z|X)}\left[\log \frac{p_\theta(X,z)}{q_\phi(z|X)}\right] \simeq \nabla_\theta \log p_\theta(X,z) \tag{6}$$

works just fine. However, $\nabla_\phi \mathcal{L}_{\theta,\phi}$ more tricky, since we can't pull ∇_ϕ into $\mathbb{E}_{q_\phi(z|X)}$.

Solution: Use reparametrization trick. Generate samples from $z \sim q_\phi(z|x)$. Reparametrize $z = g(\epsilon, X, \phi)$, where ϵ has independent distribution, e.g. $p(\epsilon) = \mathcal{N}(0,1)$. Then $\mathbb{E}_{q_\phi(X)} = \mathbb{E}_{p(\epsilon)}$ and we get a new Monte Carlo estimator

$$\nabla_\phi \mathcal{L}_{\theta,\phi} = \nabla_\phi \mathbb{E}_{p(\epsilon)} = \mathbb{E}_{p(\epsilon)}\left[\nabla_\phi \log \frac{p_\theta(X,z)}{q_\phi(z|X)}\right] = \nabla_\phi \mathbb{E}_{p(\epsilon)}\left[\log p_\theta(X|z) + \log \frac{p(z)}{q_\phi(z|X)}\right] \tag{7}$$

KL divergence

How does the KL divergence for two Gaussians look like?

$$D_{KL}(\mathcal{N}_0||\mathcal{N}_1) = \frac{1}{2}\left(\text{tr}(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu) - k + \ln\left[\frac{\det\Sigma_1}{\det\Sigma_0}\right]\right) \tag{8}$$

Hence, for $k = 2$ latent dimensions and $\mathcal{N}_0 = q_\phi(z|X) = \mathcal{N}(\mu_\phi, \sigma_\phi)$ and $\mathcal{N}_1 = p(z) = \mathcal{N}(0, I)$, we simplify

$$J_{KL} = -D_{KL} = -\sigma_\phi^2 - \mu_\phi^2 + 1 + \log(\sigma_\phi^2) \tag{9}$$

In total:

$$J = J_{\text{cross entropy}} + J_{KL} \tag{10}$$

VAE

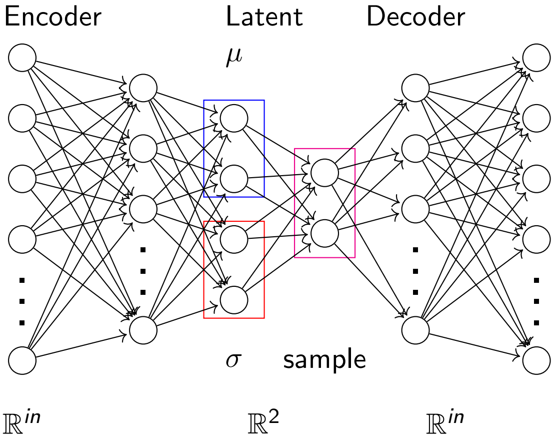

Figure: *A fully connected Variational Autoencoder with two latent dimensions.*

VAE MNIST architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 64)        640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 4100
=================================================================
Total params: 78,596
Trainable params: 78,596
Non-trainable params: 0
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 3136)              9408
_________________________________________________________________
reshape_1 (Reshape)          (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 14, 14, 64)        36928
_________________________________________________________________
conv2d_transpose_4 (Conv2DTr (None, 28, 28, 64)        36928
_________________________________________________________________
conv2d_transpose_5 (Conv2DTr (None, 28, 28, 1)         577
=================================================================
Total params: 83,841
Trainable params: 83,841
Non-trainable params: 0
_________________________________________________________________
```

Figure: *Architecture of encoder and decoder of the VAE.*

VAE MNIST visualization

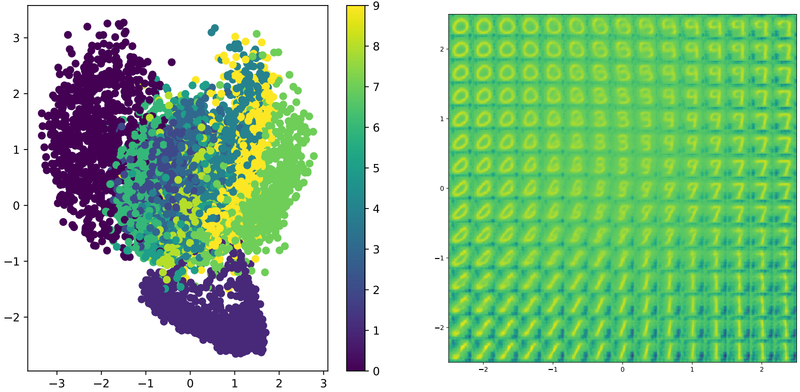

Figure: *On the left clustering of MNIST digits to two latent dimensions via encoder, on the right decoded images of samples from the two latent dimensions.*

VAE SLM visualization

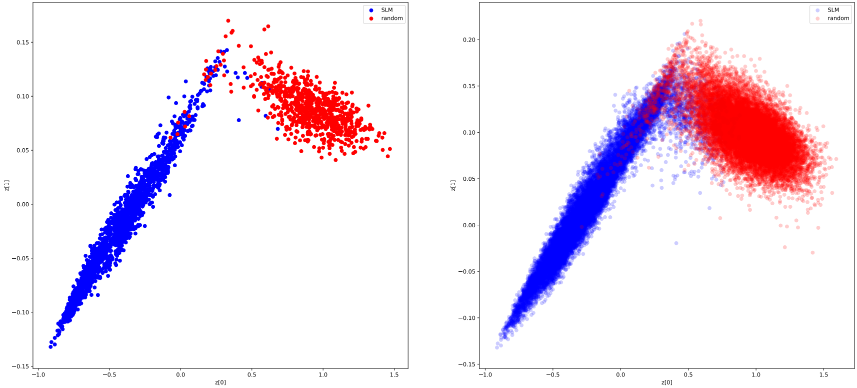

Figure: *Image of clustered Standard like models. To the left with norm $V < 5$. To the right all SLMs.*

Application: Clustering of SLM

Clustering of SLMs

- Compactification data is usually given in terms of integer matrices
- SLM from heterotic on orbifolds [1811.05993]
- SLM from $E_8 \times E_8$ on CICY [2003.13339]
- SLM from $SO(32)$ on CICY [2003.11880]

Application: Clustering of SLM

Clustering of SLMs

- Compactification data is usually given in terms of integer matrices
- SLM from heterotic on orbifolds [1811.05993]
- SLM from $E_8 \times E_8$ on CICY [2003.13339]
- SLM from $SO(32)$ on CICY [2003.11880]

To do list:

1. Create data
2. Go down to two latent dimension to visualize
3. Find clusters of SLM
4. Profit.