Autoencoders

Santiago VELASCO-FORERO
http://cmm.ensmp.fr/ velasco/

MINES ParisTech
PSL Research University
Center for Mathematical Morphology

Contents

Contents

Supervised Learning

Given a labeled dataset $(\mathbf{X}, \mathbf{Y})$, we would like to learn a mapping from data space to label space.

$\mathbf{x}$ $\xrightarrow{\text{Classification}}$ $\hat{\mathbf{y}} \in \mathbb{Z}$

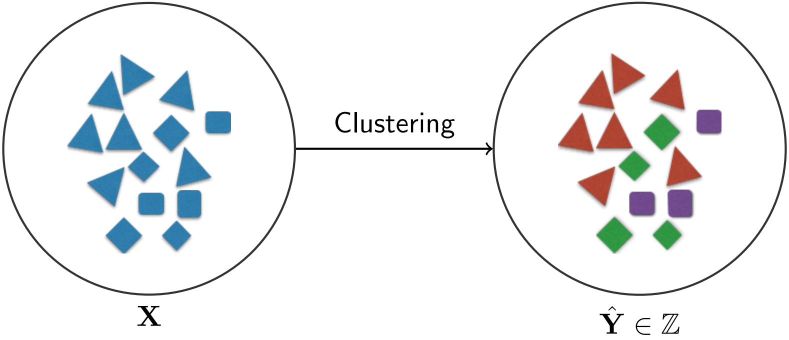
What animal is this?

$\mathbf{x}$ $\xrightarrow{\text{Regression}}$ $\hat{\mathbf{y}} \in \mathbb{R}$

How cold is it?

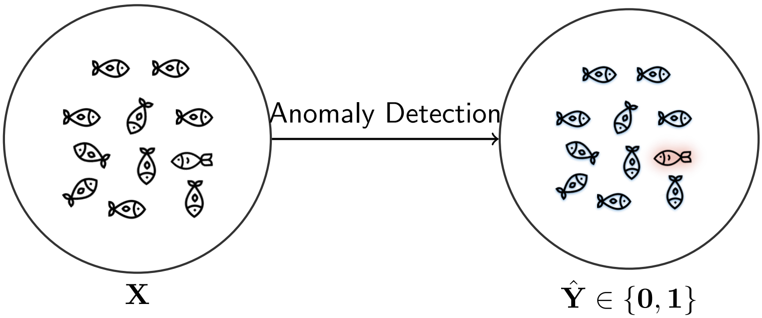
Unsupervised Learning: Clustering

Given an unlabeled dataset $(\mathbf{X})$, we would like to learn: How to group objects into similar categories?



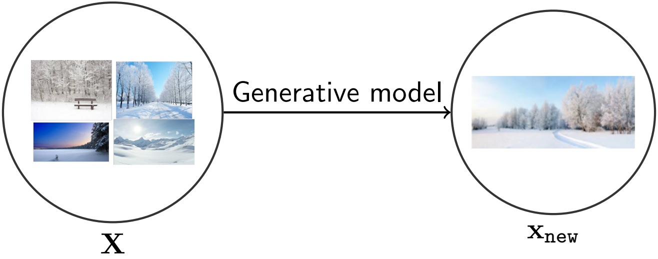
Unsupervised Learning: Anomaly detection

Given an unlabeled dataset $(\mathbf{X})$, we would like to learn: How to identify observations differing significantly from the majority of data?



Unsupervised learning: Generative Models

Given an unlabeled dataset $(\mathbf{X})$, we would like to learn: How to generate a new observations from the same distribution (unknown) of dataset?



Autoencoders

Autoencoders are neural networks whose purpose is twofold.

1. To compress some input data by transforming it from the input domain to another space, known as the *latent space* (code).
2. To take this latent representation and transform it back to the original space, such that the output is similar to the input.

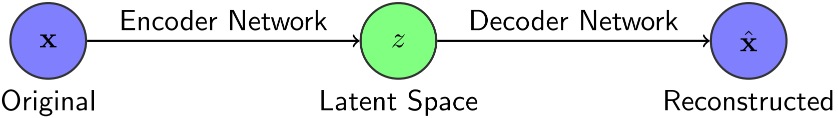


Figure: Loss function for a given input vector is usually their reconstruction error, i.e.,$L(\mathbf{x}) = (\mathbf{x} - \hat{\mathbf{x}})^2$

Contents

Autoencoder

- An *autoencoder* is a neural network that is trained to attempt to copy its input to its output.
- The network may be viewed as consisting of two parts: an encoder function $h = f(\mathbf{X})$ and a decoder that produces a reconstruction $r = g(h)$.
- If an autoencoder succeeds to learn $g(f(\mathbf{X})) = \mathbf{X}$ everywhere, then it is not especially useful (overfitting).
- The learning process can be described as minimizing the loss function:

$$L(\mathbf{X}, g(f(\mathbf{X}))), \tag{1}$$

where L is a loss function, such as mean squared error.

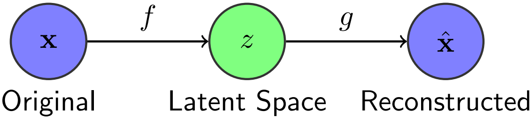


Figure: General Autoencoder structure

Over/Under complete autoencoders

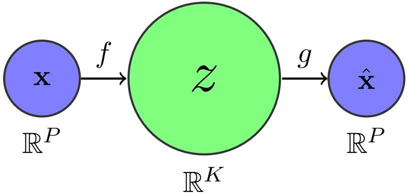


Figure: $P < K$: Overcomplete AE

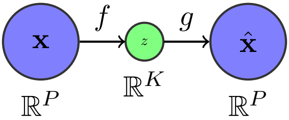


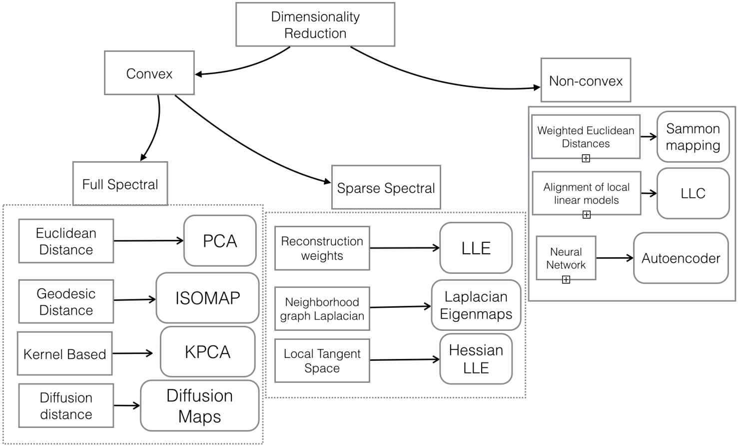
Figure: $K < P$: Undercomplete AE

An autoencoder whose code dimension is less than the input dimension is called `undercomplete`.

Contents

Dimensional reduction methods

When the decoder is linear and L is the mean squared error, an undercomplete autoencoder learns to span the same subspace as PCA.



Autoencoder vs general data compression methods

- Autoencoder are data-dependent
- MP3 or JPEG compression algorithm make general assumptions about "sound/images?, but not about specific types of sounds/image.
- Autoencoders are lossy.
- Autoencoders are learnt for a specific application.

Motivation: Nonlinear dimensionality reduction

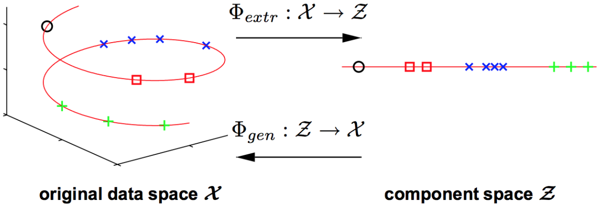


Figure: TODO: Manifold Learning

Contents

Type of Autoencoders:

1. Vanilla autoenconder
2. Regularized autoencoder (Sparse)
3. Denoising autoenconder
4. Contractive autoenconder
5. Variational autoenconder

Regularized Autoencoders

A sparse autoencoder is simply an autoencoder whose training criterion involves a sparsity penalty $\Omega(h)$ on the code layer h, in addition to the reconstruction error:

$$L(\mathbf{X}, g(f(\mathbf{X}))) + \Omega(h) \tag{2}$$

- L_1 : Cost function = Loss Function $+\frac{\lambda}{2m}\sum ||w||$
- L_2 Cost function = Loss Function $+\frac{\lambda}{2m}\sum ||w||^2$

```
from keras import regularizers
model.add(Dense(64, input_dim=64,
                kernel_regularizer=regularizers.l2(0.01)
```

Figure: Regularizers in Keras. There are Kernel and Bias Regularizer

Sparse Autoencoders

Regularization of the representation learned by the Auto-Encoders.

- Enforcing most code coefficients to be close to 0 (to be inactive).
- Capturing a more robust representation of the manifold structure.

Common implementation

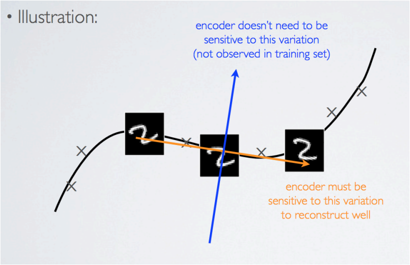
- Adding a sparsity regularizer loss to the autoencoder loss function.
- Various sparsity regularizers.
- Other existing methods:

Contractive Autoencoders

The *contractive autoencoder* introduces a regularizer on the code $h = f(\mathbf{X})$, encouraging the derivatives of f to be as small as possible:

$$\Omega(h) = \lambda \left\| \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} \right\|_F^2 \tag{3}$$

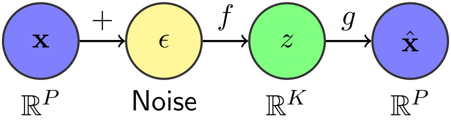
TODO



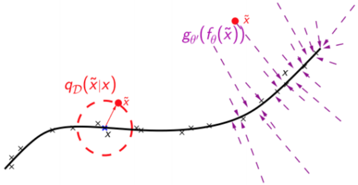
Denoising Autoencoders (DAE)

$$L(\mathbf{X}, g(f(\tilde{\mathbf{X}}))), \tag{4}$$

where $\tilde{\mathbf{X}}$ is a corrupted copy of $\mathbf{X}$ by some form of noise. An overcomplete autoencoder with high capacity can end up learning an identity function where input=output. Add noise to



Interpretation: Manifold Learning



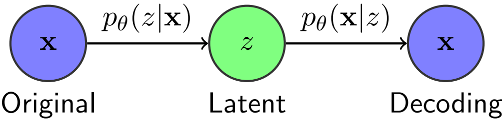
- training data lies nearby a low-dimensional manifold.
- a corrupted example is obtained by applying a perturbation of original example
- The model should learn to *project them back* to the manifold

TODO

Modern Autoencoder

- Modern autoencoders have generalized the idea of an encoder and a decoder beyond deterministic functions to stochastic mappings $p_{\texttt{encoder}}(h|\mathbf{X})$ and $p_{\texttt{decoder}}(\mathbf{X}|h)$.

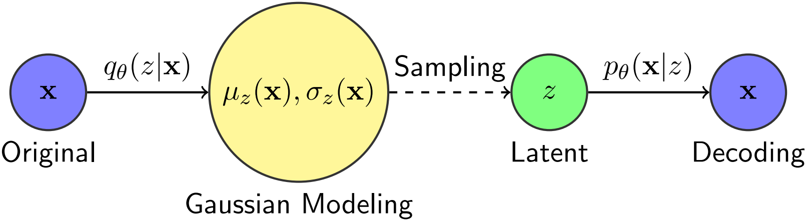
Variational Autoencoder



- Training via maximum likelihood of $p(\mathbf{x})$
- Intractability: the true posterior density $p_\theta(z|\mathbf{x})$ can be calculated
- Solutions: a) MCMC (too costly) b) Approximate $p(z|\mathbf{x})$ by means of $q(z|\mathbf{x}) = \mathcal{N}(z; \mu_z(x), \sigma_z(x))$

Contents

Variational Autoencoder



- Gaussian modeling (diagonal covariance matrix to avoid problems in high dimensionality)
- Training via maximum likelihood of $p(\mathbf{x})$
- Learning the parameters θ's via backpropagation?

Training via maximum likelihood

Assume we would like to compute the likelihood of an image $\mathbf{x}$ from the training set:

$$
\begin{aligned}
\mathcal{L}(\mathbf{x}) &= \log(p(\mathbf{x})) \\
&= \sum_z q(z|\mathbf{x}) \log(p(\mathbf{x})) \\
&= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z,\mathbf{x})}{p(z|\mathbf{x})}\right) \\
&= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z,\mathbf{x})q(z|\mathbf{x})}{q(z|\mathbf{x})p(z|\mathbf{x})}\right) \\
&= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z,\mathbf{x})}{q(z|\mathbf{x})}\right) + \sum_z q(z|\mathbf{x}) \log\left(\frac{q(z|\mathbf{x})}{p(z|\mathbf{x})}\right) \\
&= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z,\mathbf{x})}{q(z|\mathbf{x})}\right) + \underbrace{D_{KL}(q(z|\mathbf{x}), p(z|\mathbf{x}))}_{\text{how good is the approximation.}} \\
&= \mathcal{L}^{lvb}(\mathbf{x}) + D_{KL}(q(z|\mathbf{x}), p(z|\mathbf{x})) \\
&\geq \mathcal{L}^{lvb}(\mathbf{x})
\end{aligned}
$$

TODO CHECK

$$
\begin{aligned}
\mathcal{L}(\mathbf{x}) \geq \mathcal{L}^{lvb}(\mathbf{x}) &= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z,\mathbf{x})}{q(z|\mathbf{x})}\right) \\
&= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(\mathbf{x}|z)p(z)}{q(z|\mathbf{x})}\right) \\
&= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(\mathbf{x}|z)}{q(z|\mathbf{x})}\right) + \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z)}{q(z|\mathbf{x})}\right) \\
&= \mathbb{E}_{q(z|\mathbf{x})} \log(p(\mathbf{x}|z)) - D_{KL}(q(z|\mathbf{x}), p(z)) \\
&= \underbrace{\mathbb{E}_{q(z|\mathbf{x})} \log(p(\mathbf{x}|z))}_{\texttt{Expected Reconstruction}} - \underbrace{D_{KL}(q(z|\mathbf{x}), p(z))}_{\texttt{Regularization } \mathcal{N}(0,1)}
\end{aligned}
$$

- First term implies the use of many realization of sampling process (in practice we only a few of samples per training example!)
- Second term is simply a formula for diagonal multivariate Gaussian distribution.

Reparametrization trick

Backpropagation is not possible through random sampling!

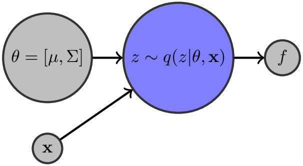


Figure: Original Formulation

Reparametrization trick

Backpropagation is not possible through random sampling!

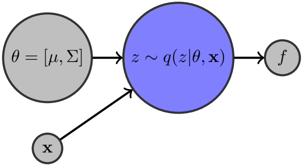


Figure: Original Formulation

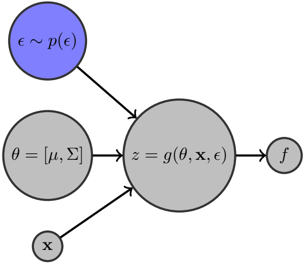
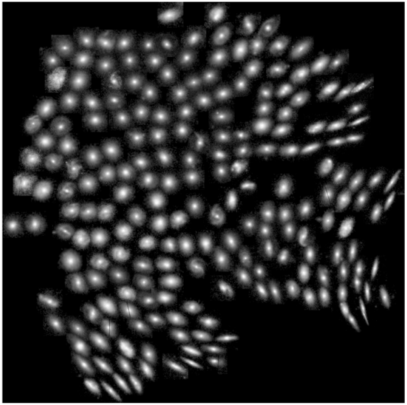


Figure: Reparametrization trick. Backpropagation

Contents

- Data denosing
- Dimensionality Reduction
- Database understanding
- Generative models

Galaxy Embedding



Galaxies embedded in two dimensions based on the means of their variational distributions, $f_\mu(x)$.

Figure: Jeffrey Regier et al, A deep generative model for astronomical images of galaxies

Applications for Autoencoders

1. Information retrieval using deep auto-encoders (semantic hashing)

KL-divergence between two multivariate Gaussian distributions

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

where k is the dimensionality of the distributions.

Contents

- Tutorial on Variational Autoencoders,CARL DOERSCH
- http://web.stanford.edu/class/cs294a/sae/sparseAutoencoderNotes.
- Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot et Yoshua Bengio, 2011.