Deep Learning

Vazgen Mikayelyan

December 12, 2020

FAST DISCOVERING THE FUTURE

Outline

KL Divergence

The KL divergence (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution:

$$KL\left(P||Q\right) = -\int_{-\infty}^{+\infty} p\left(x\right) \log \frac{q\left(x\right)}{p\left(x\right)} dx$$

KL Divergence

The KL divergence (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution:

$$KL\left(P||Q\right) = -\int_{-\infty}^{+\infty} p\left(x\right) \log \frac{q\left(x\right)}{p\left(x\right)} dx = \mathbb{E}_p \left[\log \frac{p}{q}\right],$$

where p and q are probability density functions of distributions P and Q.

KL Divergence

The KL divergence (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution:

$$KL\left(P||Q\right) = -\int_{-\infty}^{+\infty} p\left(x\right) \log \frac{q\left(x\right)}{p\left(x\right)} dx = \mathbb{E}_p \left[\log \frac{p}{q}\right],$$

where p and q are probability density functions of distributions P and Q. It can be proved that for all probability distributions P, Q, R

- $KL\left(P||Q\right) \geq 0$,

KL Divergence

The KL divergence (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution:

$$KL\left(P||Q\right) = -\int_{-\infty}^{+\infty} p\left(x\right) \log \frac{q\left(x\right)}{p\left(x\right)} dx = \mathbb{E}_p \left[\log \frac{p}{q}\right],$$

where p and q are probability density functions of distributions P and Q. It can be proved that for all probability distributions P, Q, R

- $KL\left(P||Q\right) \geq 0$,
- $KL\left(P||Q\right) = 0$ if and only if $P = Q$,

KL Divergence

The KL divergence (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution:

$$KL(P||Q) = -\int_{-\infty}^{+\infty} p(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_p \left[\log \frac{p}{q} \right],$$

where p and q are probability density functions of distributions P and Q. It can be proved that for all probability distributions P, Q, R

- $KL(P||Q) \geq 0$,
- $KL(P||Q) = 0$ if and only if $P = Q$,
- $KL(P||Q) \leq KL(P||R) + KL(R||Q)$,

KL Divergence

The KL divergence (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution:

$$KL(P||Q) = -\int_{-\infty}^{+\infty} p(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_p \left[\log \frac{p}{q} \right],$$

where p and q are probability density functions of distributions P and Q. It can be proved that for all probability distributions P, Q, R

- $KL(P||Q) \geq 0$,
- $KL(P||Q) = 0$ if and only if $P = Q$,
- $KL(P||Q) \leq KL(P||R) + KL(R||Q)$,

but there is no symmetry, i.e. $KL(P||Q) \neq KL(Q||P)$.

Jensen-Shannon Divergence

JS Divergence is the following

$$JS\left(P||Q\right) = \frac{1}{2}KL\left(P||M\right) + \frac{1}{2}KL\left(M||Q\right),$$

where $M = \frac{P+Q}{2}$.

KL Divergence for Gaussians

KL Divergence for Gaussians

Recall that probability density function (if it exists) of multivariate normal distribution with mean μ and with (non-singular, symmetric, positive definite) covariance matrix Σ is the following function:

$$f(x) = \frac{\exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}}{\sqrt{(2\pi)^k |\Sigma|}}, x \in \mathbb{R}^k.$$

KL Divergence for Gaussians

Recall that probability density function (if it exists) of multivariate normal distribution with mean μ and with (non-singular, symmetric, positive definite) covariance matrix Σ is the following function:

$$f(x) = \frac{\exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}}{\sqrt{(2\pi)^k |\Sigma|}}, x \in \mathbb{R}^k.$$

Suppose that we have two multivariate normal distributions: $\mathcal{N}_1(\mu_1, \Sigma_1), \mathcal{N}_2(\mu_2, \Sigma_2)$. Then

$$KL(\mathcal{N}_1, \mathcal{N}_2) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\mu_2-\mu_1)^{\mathsf{T}}\Sigma_2^{-1}(\mu_2-\mu_1) - k + \ln\frac{|\Sigma_2|}{|\Sigma_1|}\right).$$

KL Divergence for Gaussians

Recall that probability density function (if it exists) of multivariate normal distribution with mean μ and with (non-singular, symmetric, positive definite) covariance matrix Σ is the following function:

$$f(x) = \frac{\exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}}{\sqrt{(2\pi)^k |\Sigma|}}, x \in \mathbb{R}^k.$$

Suppose that we have two multivariate normal distributions: $\mathcal{N}_1(\mu_1, \Sigma_1), \mathcal{N}_2(\mu_2, \Sigma_2)$. Then

$$KL(\mathcal{N}_1, \mathcal{N}_2) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\mu_2-\mu_1)^{\mathsf{T}}\Sigma_2^{-1}(\mu_2-\mu_1) - k + \ln\frac{|\Sigma_2|}{|\Sigma_1|}\right).$$

In one dimensional case we will have

$$KL(\mathcal{N}_1, \mathcal{N}_2) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}.$$

Outline

What is Unsupervised Learning?

What is Unsupervised Learning?

Definition 1

Unsupervised learning is a machine learning technique that finds and analyzes hidden patterns in unlabeled data.

What is Unsupervised Learning?

Definition 1

Unsupervised learning is a machine learning technique that finds and analyzes hidden patterns in unlabeled data.

Examples?

Autoencoders

Autoencoders are neural networks that aims to copy their inputs to their outputs. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation.

Autoencoders

Autoencoders are neural networks that aims to copy their inputs to their outputs. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation. This kind of network is composed of two parts

- **Encoder**:

Autoencoders

Autoencoders are neural networks that aims to copy their inputs to their outputs. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation. This kind of network is composed of two parts

- **Encoder**:
  This is the part of the network that compresses the input into a latent-space representation.

Autoencoders

Autoencoders are neural networks that aims to copy their inputs to their outputs. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation. This kind of network is composed of two parts

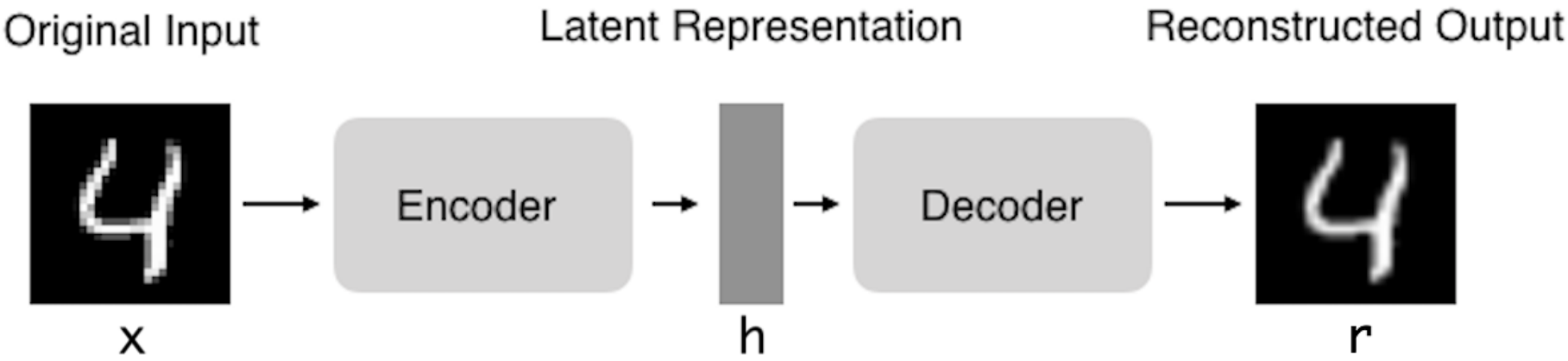
- **Encoder**:
  This is the part of the network that compresses the input into a latent-space representation.
- **Decoder**:

Autoencoders

Autoencoders are neural networks that aims to copy their inputs to their outputs. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation. This kind of network is composed of two parts

- **Encoder**:
  This is the part of the network that compresses the input into a latent-space representation.
- **Decoder**:
  This part aims to reconstruct the input from the latent space representation.

Autoencoders



What are autoencoders used for?

- Anomaly/Outlier detection.

What are autoencoders used for?

- Anomaly/Outlier detection.
- Dimensionality reduction.

What are autoencoders used for?

- Anomaly/Outlier detection.
- Dimensionality reduction.
- Data denoising.

What are autoencoders used for?

- Anomaly/Outlier detection.
- Dimensionality reduction.
- Data denoising.
- In a lot of different tasks.

Types of Autoencoders

- Vanilla Autoencoders

Types of Autoencoders

- Vanilla Autoencoders
- Multilayer/Deep Autoencoders

Types of Autoencoders

- Vanilla Autoencoders
- Multilayer/Deep Autoencoders
- Convolutional Autoencoders

Types of Autoencoders

- Vanilla Autoencoders
- Multilayer/Deep Autoencoders
- Convolutional Autoencoders
- Contractive Autoencoders

Types of Autoencoders

- Vanilla Autoencoders
- Multilayer/Deep Autoencoders
- Convolutional Autoencoders
- Contractive Autoencoders
- Regularized Autoencoders

Types of Autoencoders

- Vanilla Autoencoders
- Multilayer/Deep Autoencoders
- Convolutional Autoencoders
- Contractive Autoencoders
- Regularized Autoencoders
  - Sparse Autoencoders

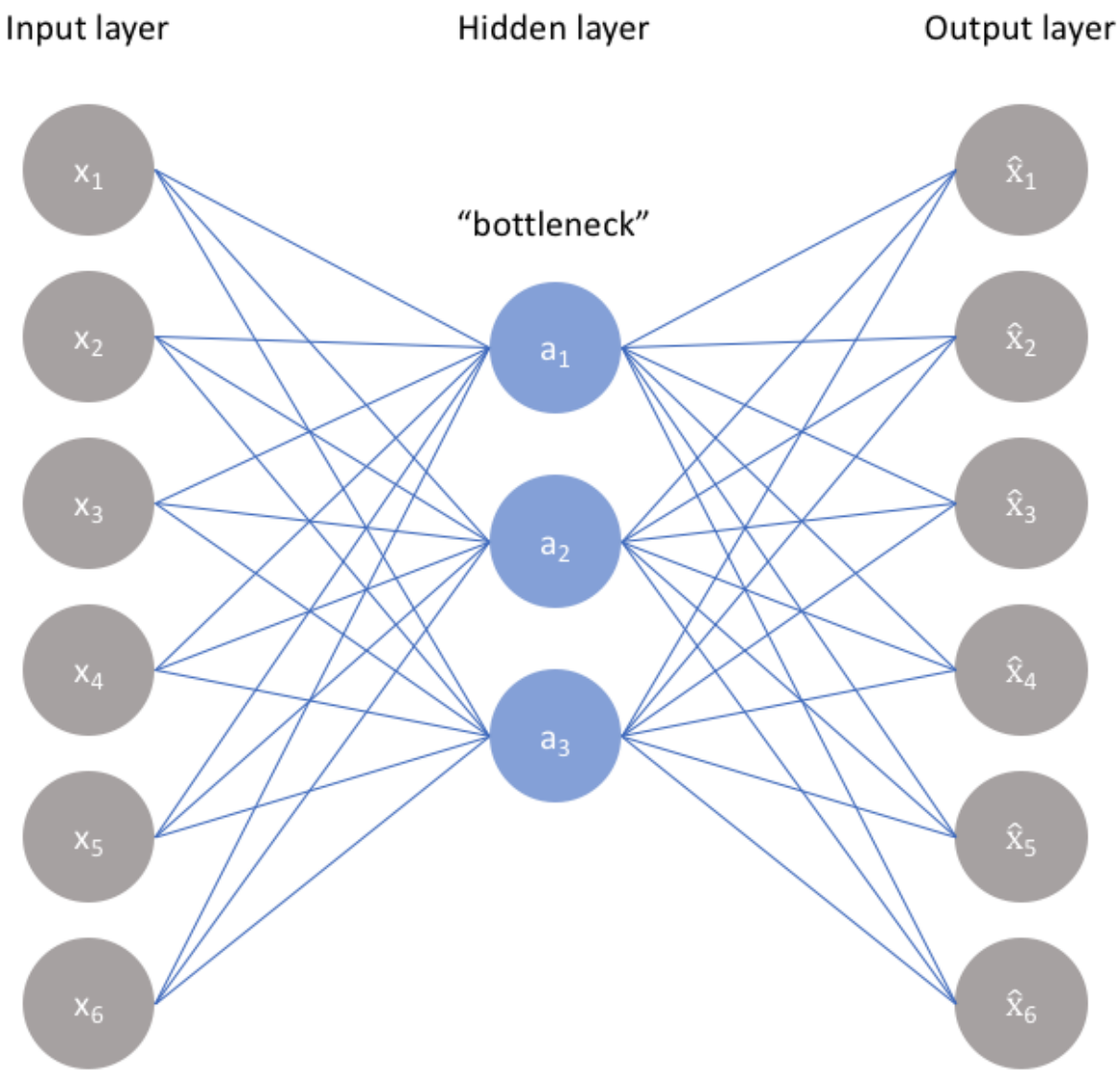
Types of Autoencoders

- Vanilla Autoencoders
- Multilayer/Deep Autoencoders
- Convolutional Autoencoders
- Contractive Autoencoders
- Regularized Autoencoders
  - Sparse Autoencoders
  - Denoising Autoencoders

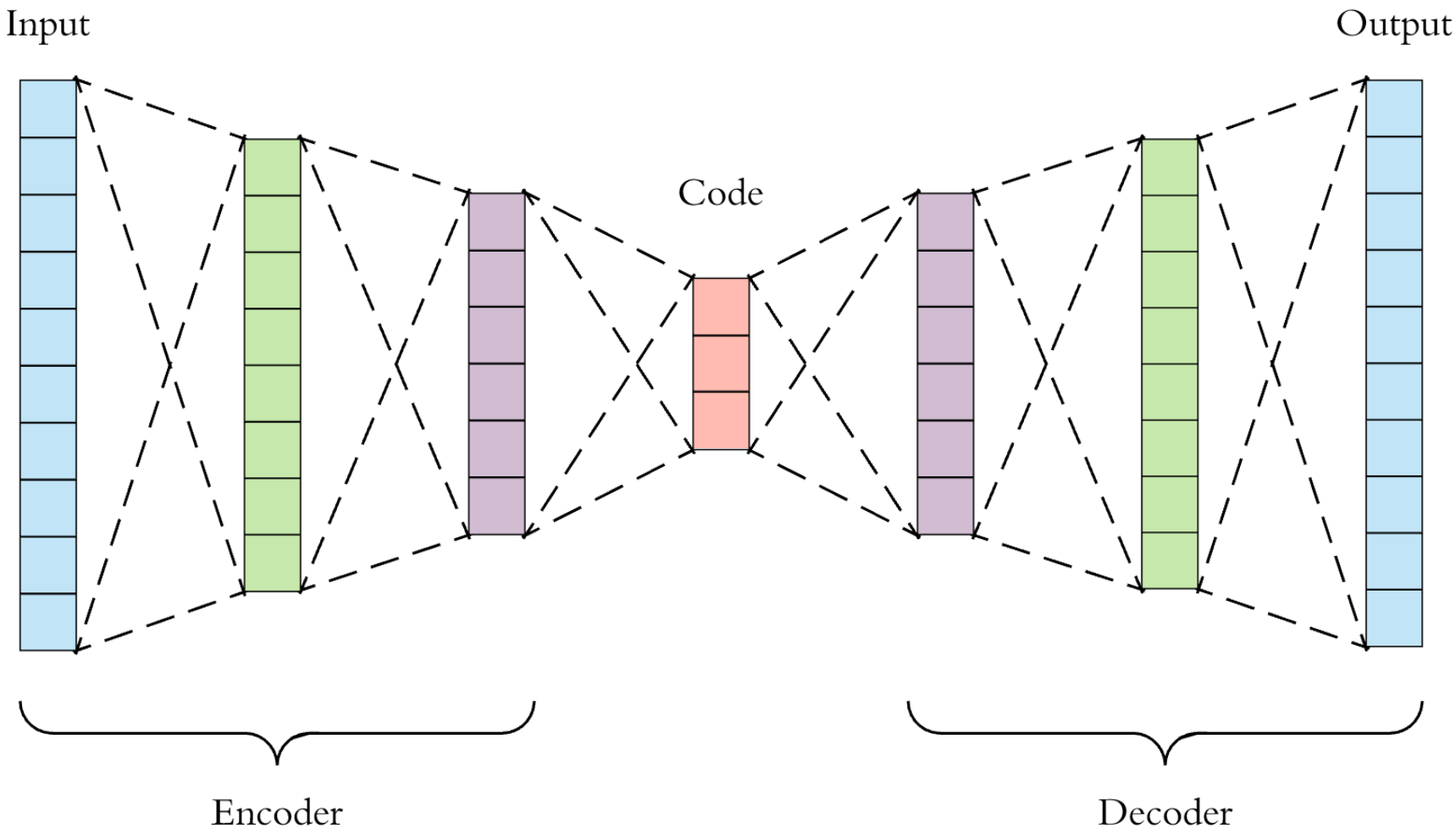
Types of Autoencoders

- Vanilla Autoencoders
- Multilayer/Deep Autoencoders
- Convolutional Autoencoders
- Contractive Autoencoders
- Regularized Autoencoders
  - Sparse Autoencoders
  - Denoising Autoencoders
- Variational Autoencoders

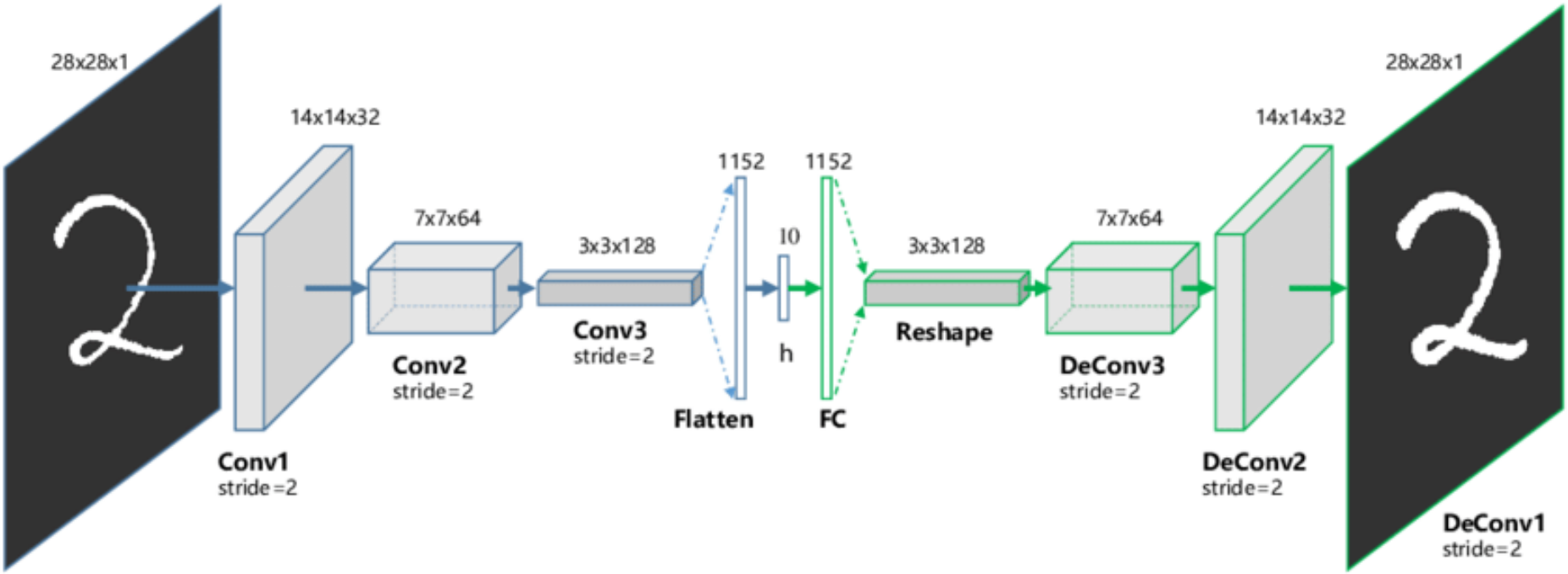
Vanilla Autoencoders



Multilayer/Deep Autoencoders



Convolutional Autoencoders



Contractive Autoencoders

- A contractive autoencoder makes this encoding less sensitive to small variations in its training dataset.

Contractive Autoencoders

- A contractive autoencoder makes this encoding less sensitive to small variations in its training dataset.
- This is accomplished by adding a regularizer, or penalty term, to whatever cost or objective function the algorithm is trying to minimize.

Contractive Autoencoders

- A contractive autoencoder makes this encoding less sensitive to small variations in its training dataset.
- This is accomplished by adding a regularizer, or penalty term, to whatever cost or objective function the algorithm is trying to minimize.
- The end result is to reduce the learned representation's sensitivity towards the training input.

Contractive Autoencoders

Let f is our encoder, g is the decoder and D is our training dataset. In the previous cases we minimize this kind of loss function:

$$\sum_{x \in D} L\left(x, g\left(f\left(x\right)\right)\right).$$

Contractive Autoencoders

Let f is our encoder, g is the decoder and D is our training dataset. In the previous cases we minimize this kind of loss function:

$$\sum_{x \in D} L\left(x, g\left(f\left(x\right)\right)\right).$$

In the case of contractive autoencoders we will minimize this one

$$\sum_{x \in D} \left(L\left(x, g\left(f\left(x\right)\right)\right) + \lambda \left\| J_f\left(x\right) \right\|_F^2 \right),$$

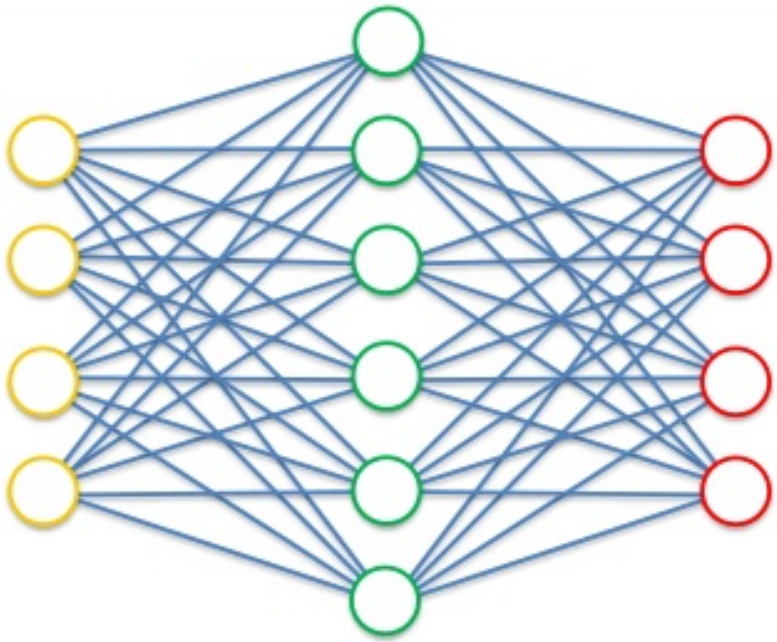
where the added summand is the square of Frobenius norm of the following Jacobian matrix:

$$\left[J_f\left(x\right)\right]_{i,j} = \frac{\partial f_j\left(x\right)}{\partial x_i}$$

i.e.

$$\left\| J_f\left(x\right) \right\|_F^2 = \sum_{i,j} \left(\frac{\partial f_j\left(x\right)}{\partial x_i} \right)^2.$$

Sparse Autoencoders



Deep Learning A-Z

© SuperDataScience

Sparse Autoencoders

- Sparse autoencoders have hidden nodes greater than input nodes. They can still discover important features from the data.

Sparse Autoencoders

- Sparse autoencoders have hidden nodes greater than input nodes. They can still discover important features from the data.
- Sparsity penalty is introduced on the hidden layer. This is to prevent output layer copy input data. This prevents overfitting.

Sparse Autoencoders

Let f is our encoder, g is the decoder and D is our training dataset, which has n samples. Denote

$$\rho_j = \frac{1}{n} \sum_{x \in D} f_j(x).$$

Sparse Autoencoders

Let f is our encoder, g is the decoder and D is our training dataset, which has n samples. Denote

$$\rho_j = \frac{1}{n} \sum_{x \in D} f_j(x).$$

We would like to (approximately) enforce the constraint $\rho_j = \rho$, where ρ is a sparsity parameter, typically a small value close to zero (say $\rho = 0.05$).

Sparse Autoencoders

Let f is our encoder, g is the decoder and D is our training dataset, which has n samples. Denote

$$\rho_j = \frac{1}{n} \sum_{x \in D} f_j(x).$$

We would like to (approximately) enforce the constraint $\rho_j = \rho$, where ρ is a sparsity parameter, typically a small value close to zero (say $\rho = 0.05$). To achieve this we will minimize the following loss function

$$\sum_{x \in D} L(x, g(f(x))) + \lambda \sum_j KL(\rho || \rho_j),$$

Sparse Autoencoders

Let f is our encoder, g is the decoder and D is our training dataset, which has n samples. Denote

$$\rho_j = \frac{1}{n} \sum_{x \in D} f_j(x).$$

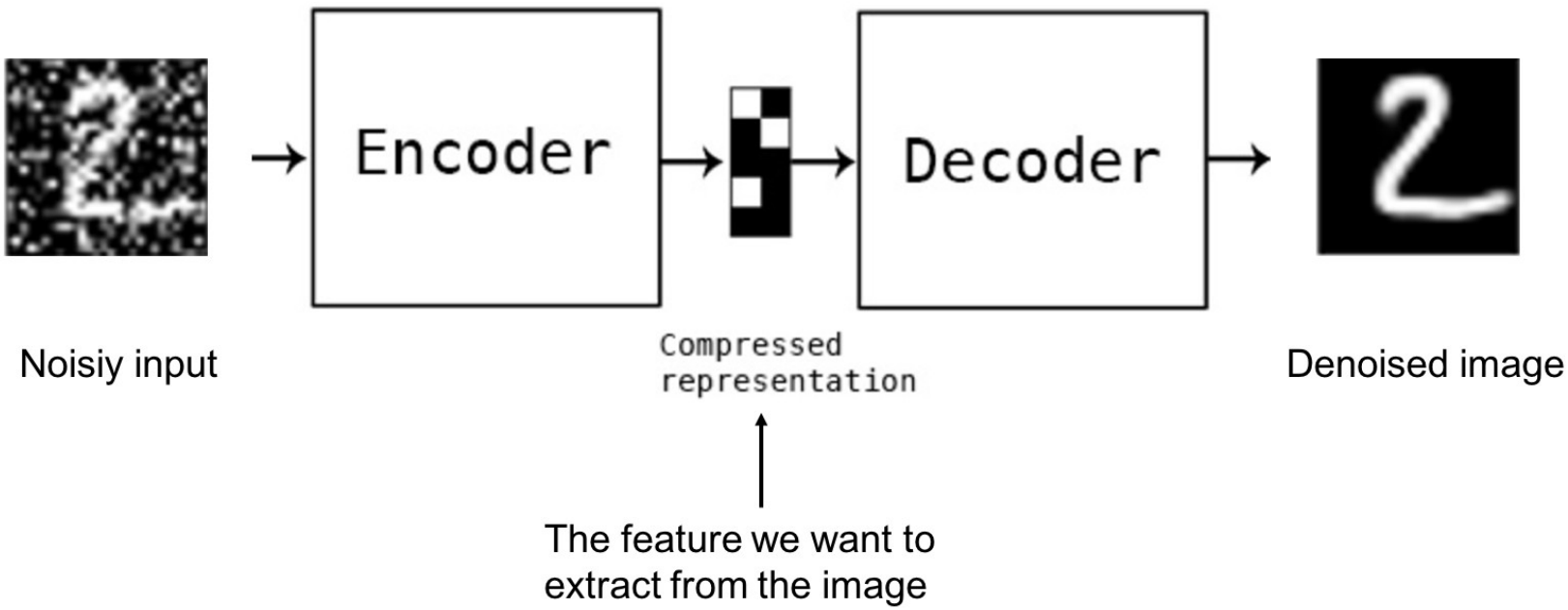
We would like to (approximately) enforce the constraint $\rho_j = \rho$, where ρ is a sparsity parameter, typically a small value close to zero (say $\rho = 0.05$). To achieve this we will minimize the following loss function

$$\sum_{x \in D} L(x, g(f(x))) + \lambda \sum_j KL(\rho || \rho_j),$$

where

$$KL(\rho || \rho_j) = -\rho \log \frac{\rho_j}{\rho} - (1 - \rho) \log \frac{1 - \rho_j}{1 - \rho}.$$

Denoising Autoencoders



Denoising Autoencoders

- Denoising autoencoders create a corrupted copy of the input by introducing some noise.

Denoising Autoencoders

- Denoising autoencoders create a corrupted copy of the input by introducing some noise.
- This helps to avoid the autoencoders to copy the input to the output without learning features about the data.

Denoising Autoencoders

- Denoising autoencoders create a corrupted copy of the input by introducing some noise.
- This helps to avoid the autoencoders to copy the input to the output without learning features about the data.
- The model learns a vector field for mapping the input data towards a lower dimensional manifold which describes the natural data to cancel out the added noise.