Advanced Recurrent Neural Networks

Anshul Thakur   Andrew Creagh   Prof. David A. Clifton

Department of Engineering Science
University of Oxford

Centre for Doctoral Training in Healthcare Innovation

Table of Contents

Table of Contents

Manifold Hypothesis

- **Manifold:** A topological space that is locally Euclidean
- **Manifold Hypothesis:** High dimensional data lie on low dimensional manifolds embedded in high-dimensional space

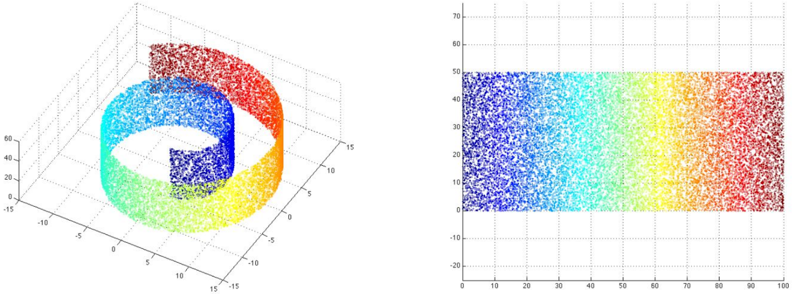


Figure: (a) A curled plane in 3-dimensional space. (b) Unrolled plane in 2-dimensional space.

Manifold Hypothesis

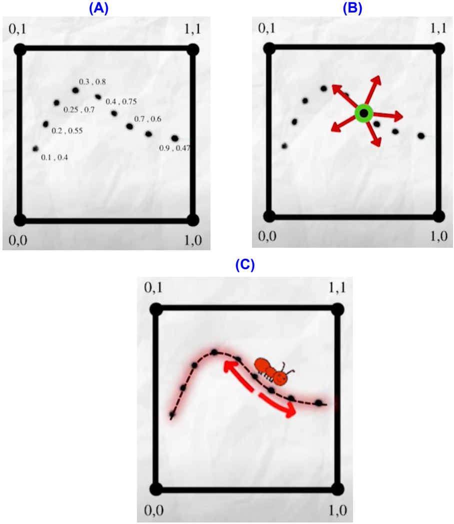


Figure: Illustration of a 1-d manifold in 2-d space.

Manifold Hypothesis

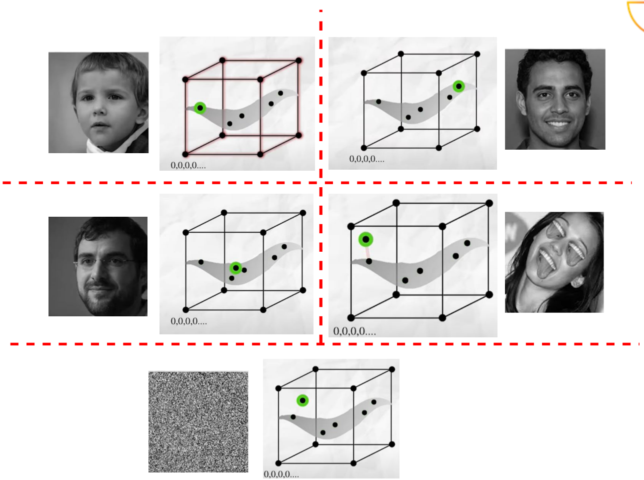


Figure: Illustration of an image manifold in image space.

Principal Component Analysis

- **Principal Components:** Orthogonal unit vectors representing variations in the data
- **Dimensionality Reduction:** Project the input N-dimensional data onto K principal components such that $K << N$

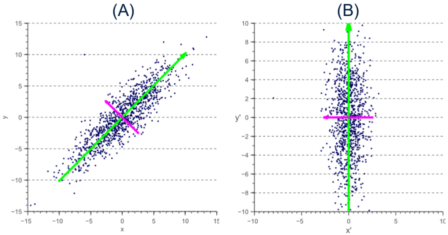


Figure: First and second principal components.

Principal Component Analysis: Algorithm

- Compute data covariance matrix:
  $$\mathbf{C} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$
- Compute eigen vectors:
  $\mathbf{C} = \mathbf{U}\Sigma\mathbf{U}^T$
  - Σ is a diagonal matrix containing eigen values
  - $\mathbf{U}$ contains the corresponding eigen vectors
- Project data on M components:
  $\mathbf{x}_p = \mathbf{U}_{1:M}^T\mathbf{x}$

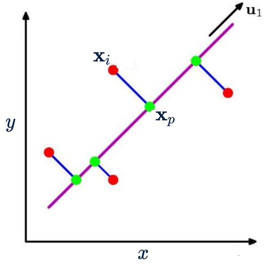


Figure: Projection on first principal component.

Principal Component Analysis: Algorithm

- **Reconstruction:** Obtaining data from the projected space to the original space:

$$\hat{\mathbf{x}} = \mathbf{X}_p \mathbf{U}_{1:M} + (\mathbf{U}_{M:D}^T \bar{\mathbf{x}}) \mathbf{U}_{M:D} \tag{1}$$

- **Objective:** Minimise $\sum_{i=1}^{N} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||_2^2$
- Optimal solution if $\mathbf{U}_{M:D}$ represents least information
- $\mathbf{U}_{M:D}$ should correspond to lesser eigenvalues

Principal Component Analysis: Eigenfaces

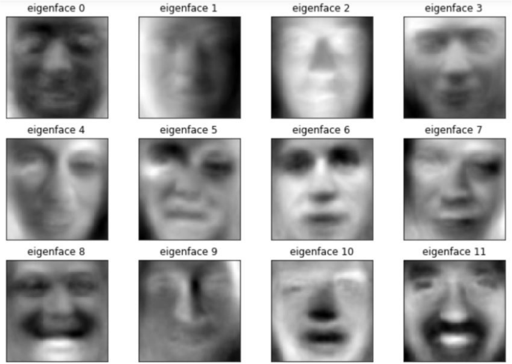


Figure: An application of PCA: Eigenfaces.

Credit: *towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184*

Principal Component Analysis is Linear!

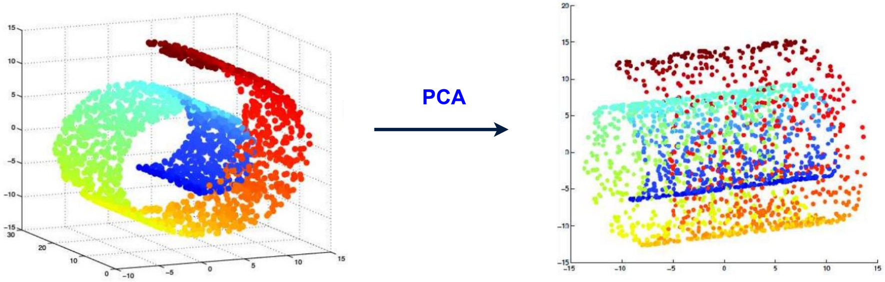


Figure: PCA on Swiss roll dataset.

Credit: *Barnabas Pozcos*

Autoencoder

- A neural network trained to reconstruct its input
- Autoencoder consists of two components:
  - **Encoder:** A function $f()$ that maps an input example $(\mathbf{x})$ to a latent representation or code or bottleneck embedding $(\mathbf{h})$
  - **Decoder:** A function $g()$ that reconstructs the input $(\mathbf{x})$ back from the code $(\mathbf{h})$

$$\mathbf{h} = f(\mathbf{x}), \qquad \hat{\mathbf{x}} = g(\mathbf{h}) \tag{2}$$

- **Loss function:** Minimise the deviation between $\mathbf{x}$ and $\hat{\mathbf{x}}$
- Not helpful if autoencoder is an identity function: $\mathbf{x} = g(f(\mathbf{x})), \forall \mathbf{x}$

Autoencoder: Illustration

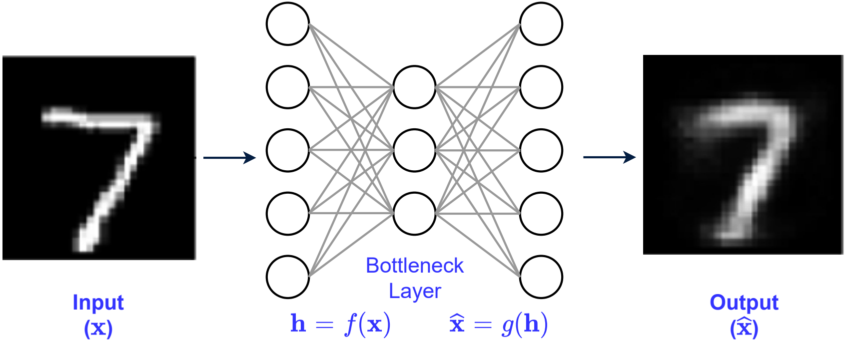


Figure: A feed-forward autoencoder.

Autoencoders: Undercomplete and Overcomplete

- **Undercomplete AE:** Dimensions of code (**h**) is less than dimensions of the input (**x**)
- Encourages the model to learn only relevant features

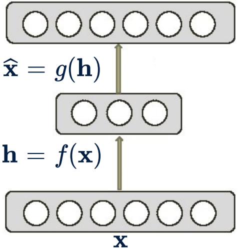


Figure: An undercomplete autoencoder.

Autoencoders: Undercomplete and Overcomplete

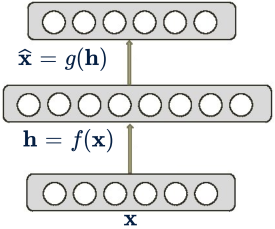


Figure: An overcomplete autoencoder.

- **Overcomplete AE:** Dimensions of code (**h**) is larger than dimensions of the input (**x**)
- No guarantee that relevant features will be extracted
- May result in AE just copying the input

Overcomplete Autoencoders

- **Sparse AE:** Requires regularisation to encourage sparsity over **h**:

$$\ell_{AE} = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2 + \lambda ||\mathbf{h}||_1 \tag{3}$$

Minimise ℓ_1 norm of **h** along with improving reconstruction

- **Contractive AE:** Regularise AE by penalising derivatives:

$$\ell_{AE} = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2 + \lambda \nabla_{\mathbf{x}} \mathbf{h} \tag{4}$$

Penalise larger changes in **h** due to small changes in **x**

Denoising Autoencoders

- Training AE to reconstruct a noise-free version ($\mathbf{x}'$) from the noisy input example ($\mathbf{x}$)
- $\hat{\mathbf{x}} = g(f(\mathbf{x}))$. Then, the loss function becomes:

$$\ell_{AE} = ||\mathbf{x}' - \hat{\mathbf{x}}||_2^2 \tag{5}$$

- No issue of identity AE as inputs and outputs are different
- Overcompleteness may help in better estimation of the noisy features

Denoising Autoencoders

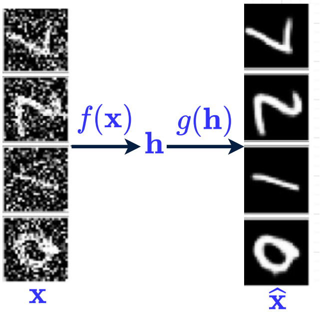


Figure: An illustration of denoising capabilities of AEs.

Denoising Autoencoders

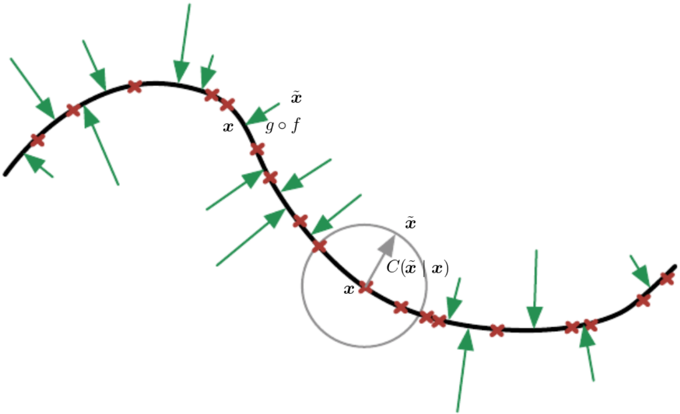


Figure: An illustration of denoising capabilities of AEs.

Credit: *Chapter 14, Deep Learning Book.*

Table of Contents

Dimensionality Reduction using AEs

- **Visualisation:** Maps high dimensional data to two or three dimensions
- **Dimensionality Reduction:** Maps high dimensional data to any lower dimensional space

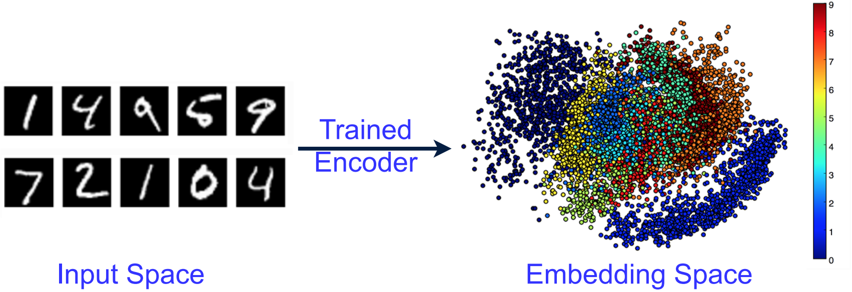


Figure: Data visualisation using trained AEs.

Segmentation using AEs

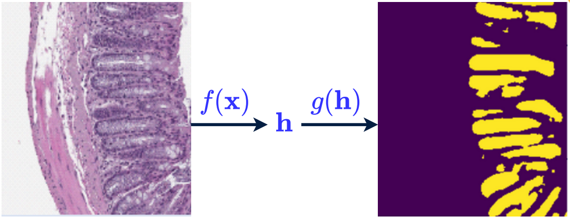


Figure: An illustration of image segmentation using AEs.

- AEs are trained to reconstruct a version of input image that only highlights the regions of interest
- Suppose $\mathbf{x}$ and $\mathbf{m}_x$ be input and mask pair. Then, the loss function for training AE is of the following form:

$$\ell_{AE} = ||g(f(\mathbf{x})) - \mathbf{m}_x||_2^2 \tag{6}$$

Unsupervised Learning

- Latent embedding or bottleneck features are semantically rich
- Latent embedding can be used for semantic clustering
- The principle of autoencoding can be used for pre-training neural networks
  - Usually helpful in case of labelled data scarcity
- Latent embedding can be used as predictive features

Table of Contents

Autoencoders for Time-series

- **Sequence-to-sequence AE:**
  - Encoder transforms a time-series to the latent vector
  - Decoder converts the latent representation to the input time-series
- **Conditioned seq2seq AEs:**
  - Generation/Decoding at next time-step is explicitly conditioned on the previous step
  - Previous time-step prediction is given as input to the current step
- **Unconditioned seq2seq AEs:** No explicit conditioning

Sequence-to-sequence AE: No Teaching

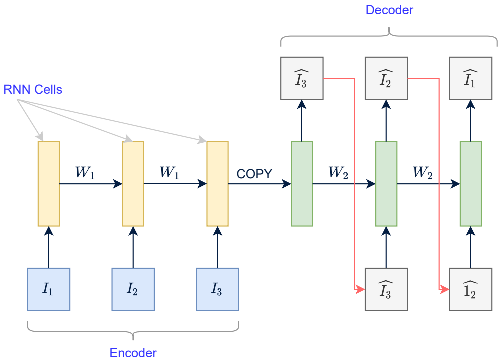


Figure: Recurrent Autoencoder Model. $\{\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3\}$ are the vectors at three steps of the time-series.

Credit: *Unsupervised Learning with LSTMs, Srivastava et. al*

Sequence-to-sequence AE: Teaching

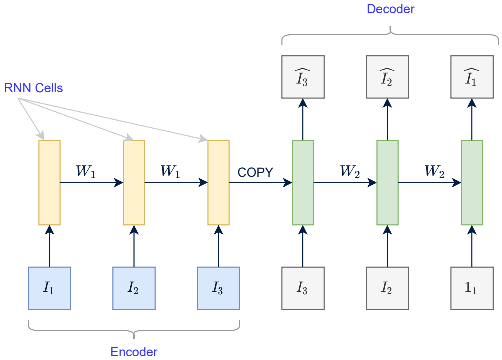


Figure: Seq2Seq Model. $\{\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3\}$ are the vectors at three steps of the input time-series $\mathbf{I}$.

Credit: *Unsupervised Learning with LSTMs, Srivastava et. al*

Neural Machine Translation

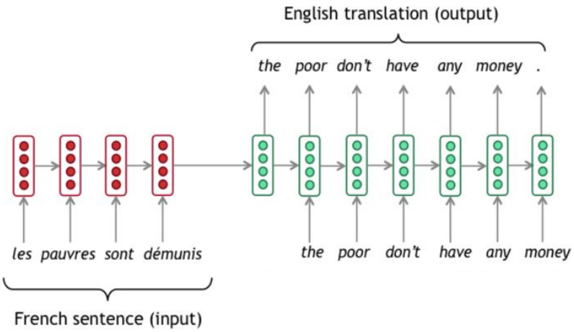


Figure: Neural machine translation using Seq2Seq encoder-decoder model.

Sequence-to-sequence AE

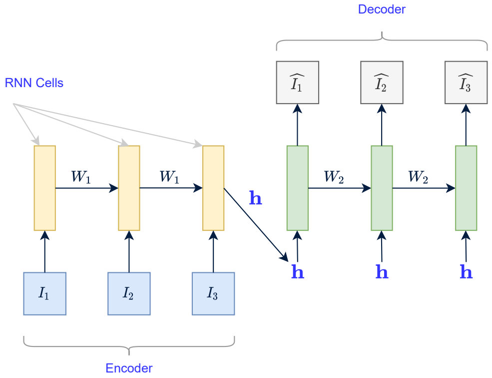


Figure: Recurrent Autoencoder Model. $\{\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3\}$ are the vectors at three steps of the time-series.

Encoder Setup: AEs for Time-Series

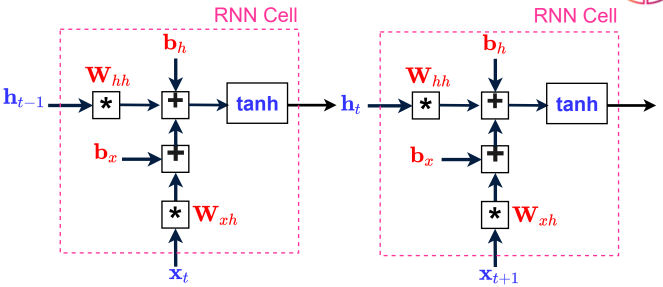


Figure: RNN cell. Biases are optional.

- $\mathbf{h}_t = \text{RNNCell}(\mathbf{x}_t, \mathbf{h}_{t-1}) = \tanh(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_x)$
- $\mathbf{h}_0 = \mathbf{0}$

Decoder Setup: AEs for Time-Series

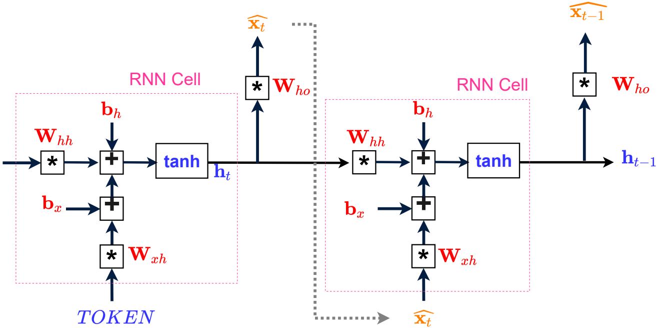


Figure: RNN based decoder setup in sequence-to-sequence AE.

- $\hat{\mathbf{x}}_{t-1} = \mathbf{W}_{ho}.\text{RNNCELL}(\hat{\mathbf{x}}_t, \mathbf{h}_t)$

Loss Function: AEs for Time-Series

- **Mean squared error:** $\mathcal{L}(\mathbf{I}, \hat{\mathbf{I}}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{I}_i - \hat{\mathbf{I}}_i)^2$
- **Mean absolute error:** $\mathcal{L}(\mathbf{I}, \hat{\mathbf{I}}) = \frac{1}{N}\sum_{i=1}^{N}|\mathbf{I}_i - \hat{\mathbf{I}}_i|$

Future Predictor: Sequence-to-sequence AE

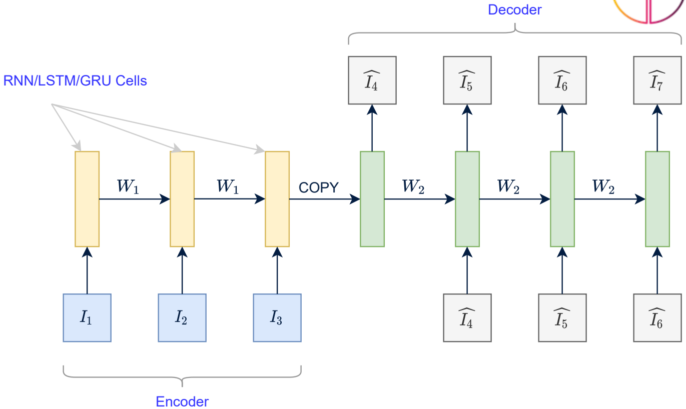
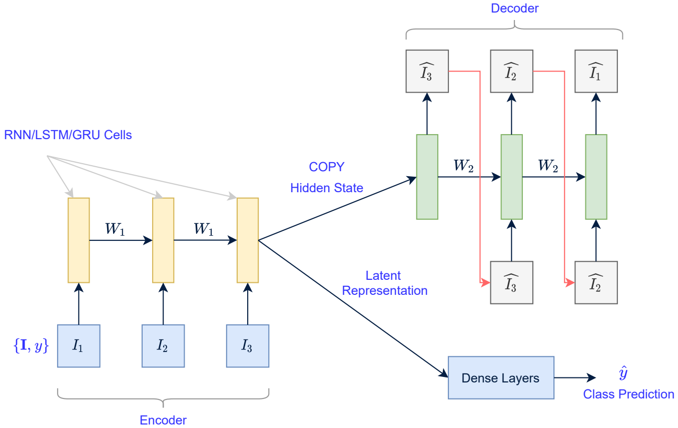


Figure: Predicting values at future time-steps.

Credit: *Unsupervised Learning with LSTMs, Srivastava et. al*

Auxiliary tasks: Sequence-to-sequence AE

- **Auxiliary tasks** can help in learning the **main tasks**
- For example: Classification and Reconstruction of time-series



$$\mathscr{L}_{AE} = \alpha.\, L(\hat{y}, y) + \beta.\, L(\mathbf{I}, \hat{\mathbf{I}})$$

Figure: A time-series classification setup with an auxiliary reconstruction task.

Skip Connections and Sparsely Connected RNNs

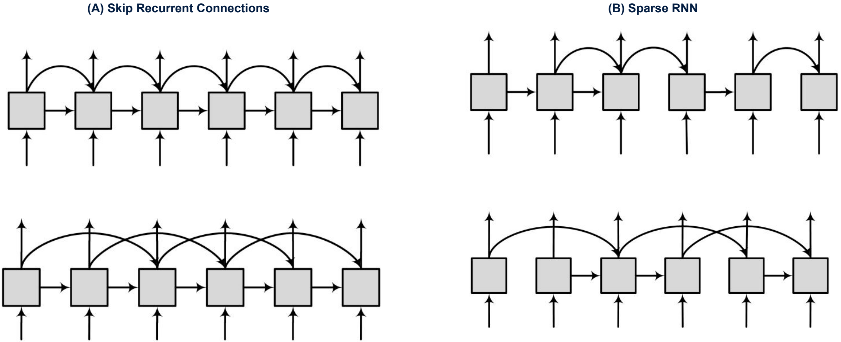


Figure: Skip connections and sparse recurrent connections between recurrent units.

Credit: *Outlier Detection for Time Series with Recurrent Autoencoder Ensembles, Kieu et. al (2019)*

Case study: Denoising using Recurrent AEs

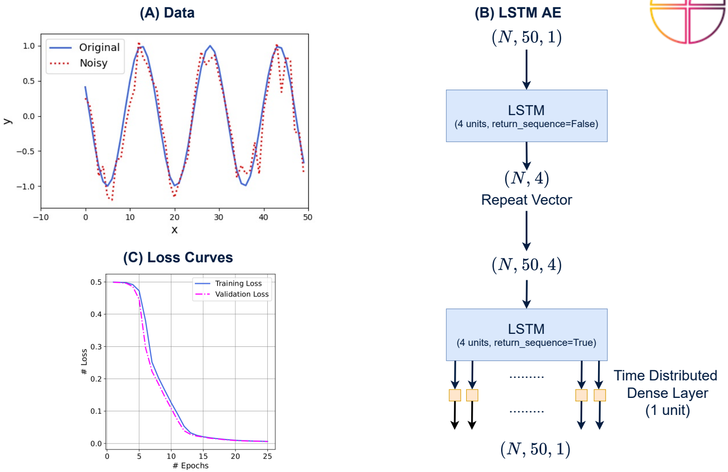


Figure: Training Recurrent AE to denoise sine waves. Means square error is used as the loss function.

Case study: Denoising using Recurrent AEs

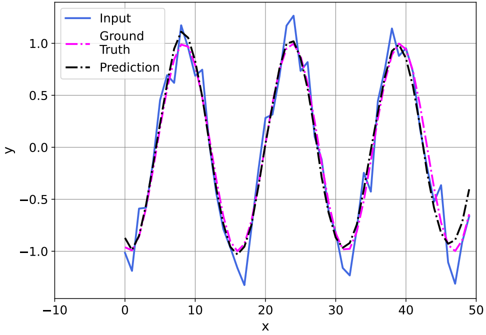


Figure: Denoising using trained Recurrent AE.

Table of Contents

Generative vs Discriminative Models

- **Discriminative models:** Capture conditional probability $p(y|\mathbf{x})$
  - Tells us how likely a label is to apply to an instance
- **Generative models:** Capture $p(\mathbf{x}, y)$ or $p(\mathbf{x})$ if no labels
  - Tells us how likely a given instance is

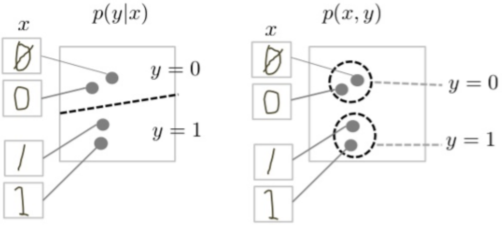


Figure: Discriminative vs generative modelling of hand-written digits.

Credit: `developers.google.com/machine-learning/gan/generative`

Autoencoder as generative model?

- **Autoencoder:** Encoder maps an input example $(\mathbf{x})$ to the latent embedding $(\mathbf{h})$ and a decoder maps this embedding back to the input space
- How will the reconstructed example look like if we use $\mathbf{h} + \delta$ to reconstruct $\mathbf{x}$?
- If $\mathbf{h} + \delta$ is on manifold, we will be fine
- Any idea about distribution of $\mathbf{h}$ can allow us to generate meaning examples in input space

Variational Autoencoder

- Learns a distribution over latent space
- Samples a hidden embedding from this distribution
- Use decoder to map the sampled embedding to input space

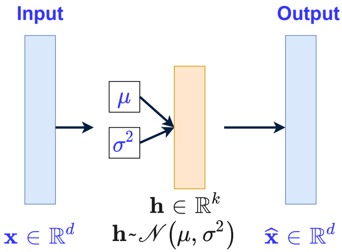


Figure: An illustration of basic VAE framework.

Variational Autoencoder

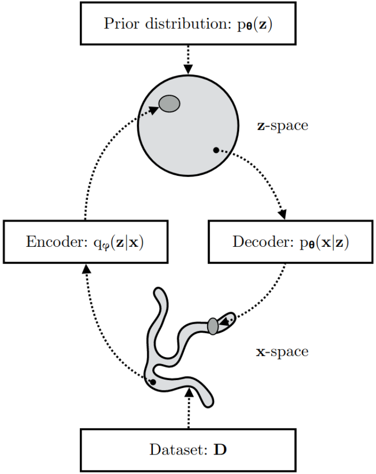


Figure: An illustration of basic VAE framework.

Evidence Lower Bound (ELBO)

- Given $q_\phi(\mathbf{z}|\mathbf{x})$, the log likelihood can be represented as:

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right]}_{\substack{=\mathcal{L}_{\boldsymbol{\theta},\phi}(\mathbf{x}) \\ \text{(ELBO)}}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right]}_{=D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))}
\end{aligned}
$$

- $\mathcal{L}_{\theta,\phi} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] = \log p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$
- $\mathcal{L}_{\theta,\phi} \leq \log p_\theta(\mathbf{x})$

Variational Autoencoder

- Let $q_\phi()$ be the encoder and p_θ be the decoder
- **Loss function:** $\mathcal{L}(\phi, \theta) = \sum_{i=1}^{N} l_i(\phi, \theta)$
- $l_i = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}_i)} \left[\log p_\theta(\mathbf{x}_i|\mathbf{z})\right] + \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}_i)||p(\mathbf{z})\right)$
- First term encourages better reconstruction
- Second term acts as regulariser: Produces $\mathbf{z}$ that follow normal distribution

Reparameterisation Trick

- Stochastic sampling is non-differentiable
- $\mathbf{z} \sim q_\phi(\mathbf{z}|x) = \mathcal{N}(\mu, \sigma^2)$
- $\mathbf{z} = \mu + \sigma \odot \epsilon$

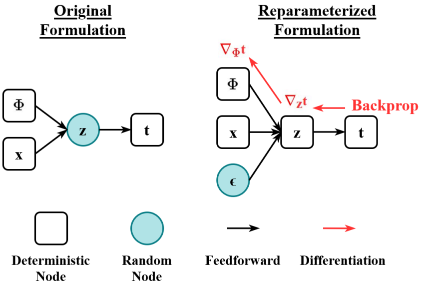


Figure: The schematic representation of reparameterisation trick.

Table of Contents

RNN-VAE for Time-series Modelling

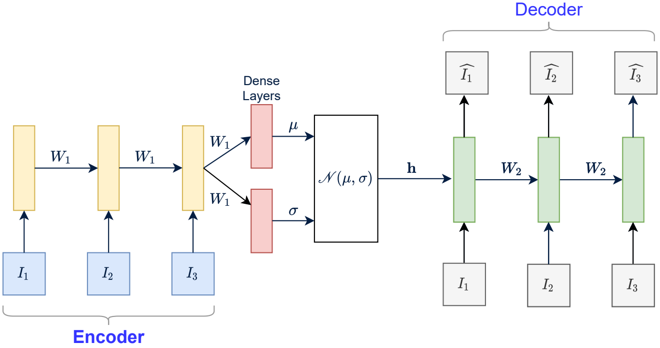


Figure: A schematic illustration of RNN-VAE.

Credit: *Latent ODEs for Irregularly-Sampled Time Series, Rubanova et. al* (2019)

What RNN-VAE can do?

- can generate EHR time-series data
- can compose music! (MusicVAE)
- can be used to generate sentences

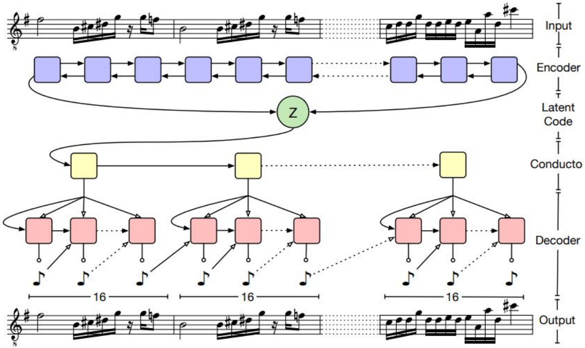


Figure: Schematic of hierarchical recurrent Variational Autoencoder model, MusicVAE.

Credit: *A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music, Roberts et. al (2019)*

Table of Contents

Further Reading

- Transformers
- Conv1d architectures for seq2seq modelling: WavNet
- Generative adversarial networks for time-series generation
- Neural ODE for continuous modelling of RNN hidden dynamics