Deep Learning in Finance

week 5: autoencoders

Damien Challet
damien.challet@centralesupelec.fr

3rd December 2024

So far

$$\text{DL} \equiv (\text{input nature, architecture, learning, output, loss})$$

input $\Longrightarrow$ architecture

architecture $\equiv$ structure$+$weights$+$activation functions

weights and structure $\Longrightarrow$ representation

$$\text{input} \rightsquigarrow \text{representation} \rightsquigarrow \text{output}$$

Model fitting

- Input data X
- Minimize cost function $\rightarrow$ parameters θ
  - MaxLik
  - (nonlinear) Least squares
  - ...
- Compressed information about system:

(model,parameters)

$\equiv$ representation of the system

Model fitting: least squares

- Fit data $X \in R^{\mathrm{N} \times T}$
- Model $\mathcal{M}$
$$\hat{X} = \mathcal{M}(\theta, \xi_{external}),$$
where $\xi_{external}$ includes noise
- Find
$$\hat{\theta} = \arg\min_{\theta} E_{\xi} ||X - \mathcal{M}(\theta, \xi)||_2$$
- $\hat{\theta}$ is such that the model reproduces the best X
- Hard part: model choice, number of parameters

Model fitting, least squares and NNs

- Least squares + model

$$X, \mathcal{M}(\theta) \rightarrow \text{loss } ||\text{X-}\mathcal{M}(\theta)||_2$$

$$\nabla\text{loss}||X - \mathcal{M}(\theta)||_2 \rightarrow \hat{\theta} \text{ at fixed } \mathcal{M}$$

- NNs

$$\text{X}, NN(X) \rightarrow \text{loss } ||X - NN(X)||_2$$

$$\nabla\text{loss}||X - NN(X)||_2 \rightarrow NN$$

NN: model + parameters

→ how to fit model and parameters?

GAN: from parameters/representation to object

- draw z from $P(z)$
- NN: $x = G(z)$: realistic object
- From object to features
  - in principle
    $$G^{-1}(x) = z$$
  - in reality
    $$z = \arg\min_{\zeta} ||x - G(\zeta)||$$
  - $\rightarrow$ ~~compute~~ learn G^{-1}

How to learn good representations?

- Learn G^{-1} and G at the same time

$$x \to z = G^{-1}(x) \to G(G^{-1}(x)) = x$$

- In practice, $\hat{G}$, $\hat{G^{-1}}$

$$x \to \hat{z} = \hat{G^{-1}}(x) \to \widehat{NN}(x) = \hat{G}(\hat{G^{-1}}(x)) = \hat{\hat{x}}$$

- Learning the identity $\equiv$ minimizing loss

$$\mathcal{L} = \sum_i ||\hat{\hat{x}}_i - x_i||$$

Autoencoder

- AE $\equiv$ Deep network with bottleneck

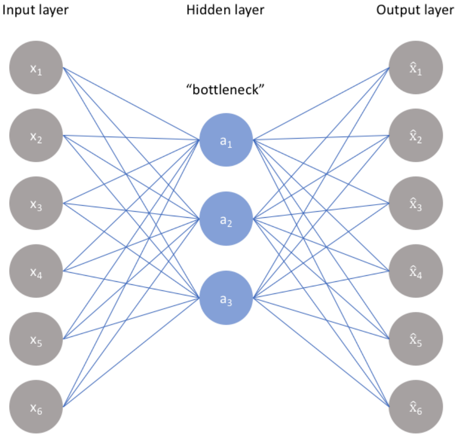
$$\text{encoder } E \rightarrow \text{ features } \rightarrow \text{ decoder } D$$

- Aim: learn the identity

$$D(E(x)) = x$$

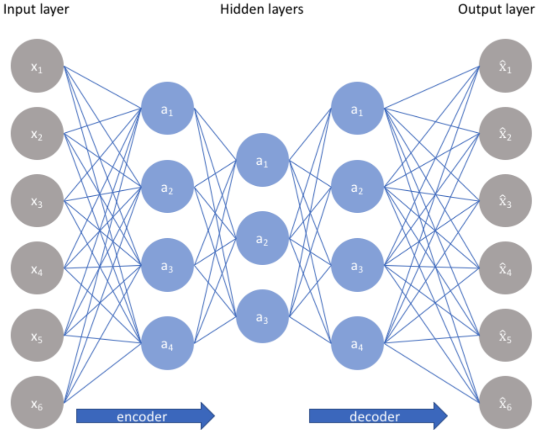
Encoder-feature-decoder

[source]



Stacked auto-encoders

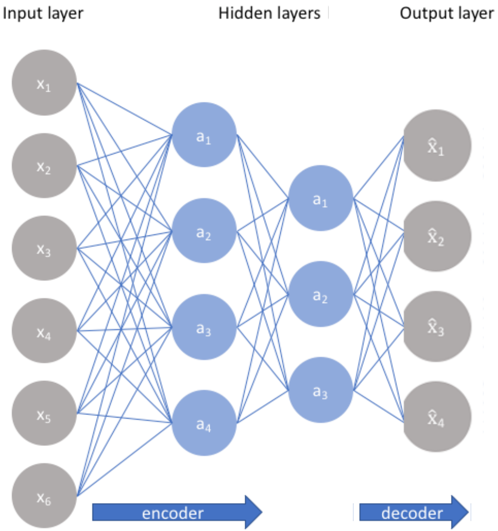
[source]



rule of thumb: learn layer by layer

Restricted auto-encoders

Partial reconstruction



N.B.: NNs must be non-linear

AE+MSE+linear activations $\equiv$ PCA

- bottleneck dimension B= projection dimension

Role of bottleneck size

B: number of neurons in the bottleneck

- $B < N$: undercomplete AO
  - $B = 1$:
    - encoder fits 1-dimensional distribution $\rightarrow$ z-score / p-value
    - decoder generalise z to input dimension
  - $B = 2$
    - able to fit 2 parameters, or joint distributions
  - $B = 3$
    - e.g. 2-dimensional averages + standard deviation
- $B \geq N$: complete/overcomplete
  - seems stupid
  - corresponds to kernel trick

Role of network complexity

- Too powerful AE, $B = 1$

  Perfect learning

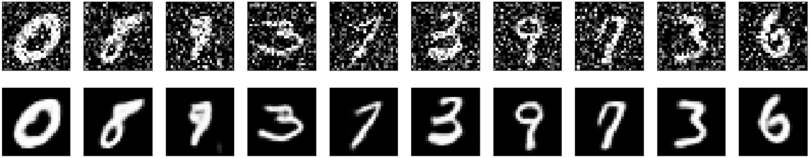
  Not good at generalizing

- Input too sharp: add noise (denoising autoencoder)

Denoising auto-encoders

- Train an AE with sharp objects
- Create noisy samples/Dropout and continue training
- Output: filtered object

Example [link]



Additional reading

Goodfellow, Bengio, Courville (2016) [link]
Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow [link]

- Tricks of the trade

- Constraints on AE:
  - symmetric weights
  - sparsity of activation in the bottleneck (L1)
  - bound gradients

Autoencoder architecture

- Images: CNN and Inverse CNN
- Timeseries
  - Dense
  - CNN
  - recurrent LSTM, GRU
  - attention

Uses of AE

1. Encoder
   - simplified information
   - (soft) clustering of z
   - compression

2. Decoder
   - generation: sample $z \to$ synthetic x

3. Full AE:
   - Denoising
   - Anomaly detection
     outlier $\to$ outlier z $\to$ outlier reconstruction loss $||\hat{\hat{x}}_i - x_i||$

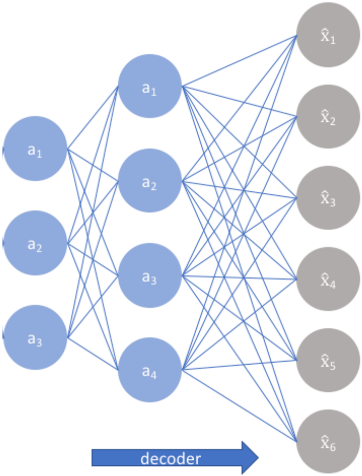
AE: encoder

- Train a (V)AE
- Compute features $z_i = \text{encoder}(x_i)$
- In feature space
  - clustering (K-means, DBscan)
  - sample similarity
- See also
  - UMAP [paper][package]
  - SNEkhorn [paper][package]

AE: decoder

- Sample z
- Decode x

  Generate $x = G(z)$
- $\equiv$ GAN



Anomaly detection

Large reconstruction error $\longleftrightarrow$ anomaly

- example : Gorduza et al (2022) [preprint]: correlation matrix reconstruction

3.1 Correlation between AUROC t+1 and volatility

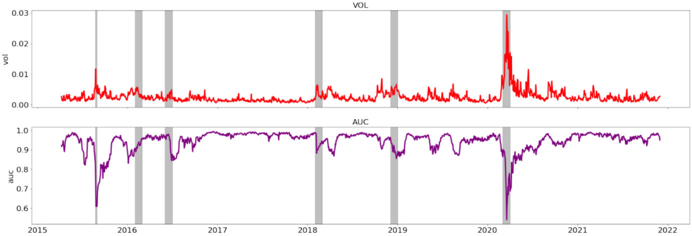


Figure 4: Time series of volatility vs $\mathbf{AUROC}_{t+1}$

- Note: may be due to correlation matrix estimation problems.

Variational Auto Encoders:

- Samples $x \in \mathbb{R}^N$ from unknown distribution $P(x)$
- How to generate new samples that look like x?
  - sample features $z \sim P(z) \to P(x, z)$
- Aim: find $P(x, z)$ that maximises

$$P(x) = \int P(x, z) dz$$

N.B.: $P(x)$ is unknown

- Equivalent problem: fix $P(z)$, find $P(x|z)$

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

Encoder, decoder

Equivalent problems: find either

1. decoder
$$P(x|z)$$
2. encoder
$$P(z|x)$$

via Bayes theorem

$$P(x|z) = \frac{P(z|x)P(x)}{P(z)}$$

DL: variational autoencoders (VAE)

Kingma and Welling (2014) [link]: learn simultaneously

1. Decoder

$$p_\theta(x|z) \simeq P(x|z)$$

2. Encoder

$$\begin{aligned} q_\phi(z|x) &\simeq P(z|x) \\ &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma} I) \text{ (hypothesis)} \end{aligned}$$

- Prior

$$p(z) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma} I)$$

- Aims:
    1. $q_\phi(z|x) \simeq p_\theta(z|x)$
    2. encoding $\rightarrow$ $P(z)$ $\rightarrow$ decoding

VAE: loss function

1. Ensure that $q_\phi(z|x) \simeq p_\theta(z|x)$

   Kullback-Leibler discrepancy: loss of information of approximating $f(y)$ with $g(y)$

   $$D_{KL}\left(f(y)||g(y)\right) = E_{f(x)}\left[\log \frac{g(y)}{f(y)}\right] = \int f(y) \log \frac{g(y)}{f(y)} dy$$

2. maximum likelihood

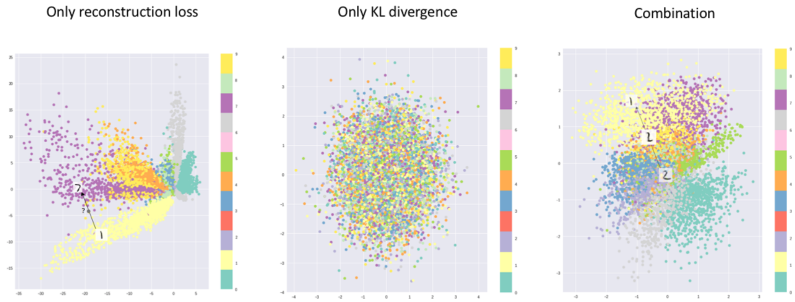
   $$\max \log P(x)$$

- Playing with discrepancy D_{KL} and Bayes rule [detailled computations]

$$\mathcal{L}(\theta, \phi) = E_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}\left(q_\phi(z|x)||p(z)\right)$$

Loss function

1. Jordan (2018) [link]: latent space



KL divergence forces $q(z|x)$ to be Gaussian with no cross-correlation

- $\beta-$VAE, $\beta > 1$, sometimes improvement

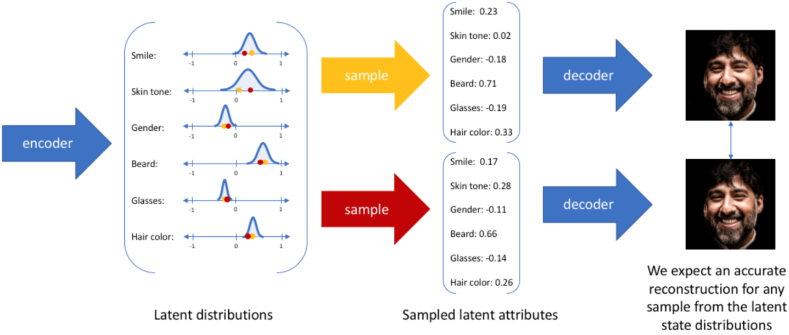
$$\mathcal{L}(\theta, \phi) = E_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \beta D_{KL}\left(q_\phi(z|x)||p(z)\right)$$

VAE: latent states

Decoder

$$
\begin{aligned}
q_\phi(z|x) &\simeq P(z|x) \\
&\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma} I) \text{ (hypothesis)}
\end{aligned}
$$

outputs $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$: multidimensional

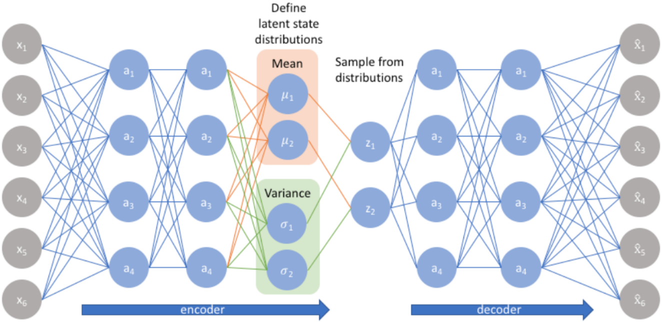


Jordan (2018) [link]

VAE: use cases

1. Clustering $\rightarrow$ prediction
2. Scenario generators
3. Anomaly detection
4. Conditional VAEs

VAE as scenario generators



Jordan (2018) [link]

- Sampling latent space according to $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma} I) \rightarrow$ scenarios

Useful references

- Goodfellow, Bengio, Courville (2016) [link]
- Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow [link]
- VAE: detailed loss function computation: [link]