Deep Generative Models (Latent Variable Models)

Aapo Hyvärinen

for the NNDL course, 25 April 2025

- Definition of deep latent variable models (DLVM)
- How to estimate an DLVM? Why maximum likelihood is difficult.
- Approximations of likelihood: Variational Autoencoders (VAE)
- Estimation of DLVM by classification: Generative Adversarial Networks (GAN)
- Problem of identifiability
- (Nonlinear) Independent Component Analysis

Literature: Prince's book Chapters 15, 17 (only part of the material)

Deep Latent Variable Models (DLVM): general definition

- Express observed data $\mathbf{x}$ as a function of latent vector $\mathbf{z}$
  - latent = hidden, i.e. unobserved
- Model (hypothetical) data generation in two stages
  - $\mathbf{z}$ is sampled from a "prior" $p(\mathbf{z})$
  - Given the sampled $\mathbf{z}$, sample $\mathbf{x}$ from the "likelihood" $p(\mathbf{x}|\mathbf{z})$
    - Terminological note: not quite the same "likelihood" as earlier
  - Formal definition:

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

    where $\boldsymbol{\theta}$ is a vector of parameters, e.g. in a neural network
- Typically, prior $p(\mathbf{z})$ is something very simple, e.g. independent z_i, and not dependent on any parameters
- Likelihood $p(\mathbf{x}|\mathbf{z})$ models some kind of "mixing", e.g.
  - Mixing of real-life "sources" in a measurement system
  - Mixing of image features when composing pixels
- *Generative model* is a term with many meanings; we use it in the same sense as a (deep) latent-variable model.

Background: covariance matrix and whiteness

- We define the *covariance matrix* as

$$[\mathrm{cov}(\mathbf{z})]_{ij} = \mathrm{cov}(z_i, z_j) = E\{z_i z_j\} - E\{z_i\}E\{z_j\} \tag{1}$$

or simply for zero-mean data:

$$\mathrm{cov}(\mathbf{z}) = E\{\mathbf{z}\mathbf{z}^T\} \tag{2}$$

- We have for any linear transformation:

$$\mathrm{cov}(\mathbf{M}\mathbf{z}) = \mathbf{M}\mathrm{cov}(\mathbf{z})\mathbf{M}^T \tag{3}$$

- We often define the z_i to be uncorrelated, and to have unit variance

$$\mathrm{cov}(\mathbf{z}) = \mathbf{I} \tag{4}$$

This is properly called *whiteness*.

- It is easy to linearly transform data to be white ("whitening")

Deep Latent Variable Models: Very basic instance

- Simplest thing is to use white Gaussian latent variables
  - Define prior $p(\mathbf{z})$ so that $\mathbf{z}$ zero-mean white Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, implying the entries z_i are mutually independent
- A mixing $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ widely used in DL: nonlinear transformation with additive Gaussian noise

$$\mathbf{x} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}) + \mathbf{n}, \tag{5}$$

  with $\mathbf{n}$ white Gaussian noise,
  $\mathbf{f}_{\boldsymbol{\theta}}$ is some nonlinear function (NN)
- $\dim(\mathbf{x})$ can be larger than $\dim(\mathbf{z})$:
  leads to dimension reduction like autoencoders

Degenerate case

- Could we omit any noise, just invertible transformation?

$$\mathbf{x} = \mathbf{f}_\theta(\mathbf{z}) \tag{6}$$

- In the case without noise, it is important that $\dim(\mathbf{x}) \leq \dim(\mathbf{z})$, otherwise the distribution of $\mathbf{x}$ is degenerate
  - i.e., totally concentrated in a lower-dimensional manifold
  - This leads to all kinds of problems: even pdf does not exist!
- An interesting approach is to assume the dimensions equal, which leads to *(normalizing) flows* (not treated here)
- For the time being, we assume dimension reduction $\dim(\mathbf{x}) >> \dim(\mathbf{z})$,and therefore noise has to be there so that the pdf exists and there is no degeneracy

Estimation of generative models by maximum likelihood

- Let's first consider maximum likelihood estimation
- But to compute the likelihood requires we *integrate out* the $\mathbf{z}$:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z} \tag{7}$$

- If we could compute this, we could compute the log-likelihood:

$$\log L(\boldsymbol{\theta}; \mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \tag{8}$$

- But there is a multidimensional integral — perhaps even worse than in EBM!

Likelihood computation in very basic model

- Considering very "simple" model considered above:

$$\mathbf{x} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}) + \mathbf{n} \tag{9}$$

  with Gaussian white $\mathbf{z}$, and $\mathbf{n}$ Gaussian of variance σ^2
- We have (assuming random vectors are all of dim d)

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z})\|^2\right) d\mathbf{z} \tag{10}$$

$$p_{\mathbf{z}}(\mathbf{z}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\|\mathbf{z}\|^2\right) \tag{11}$$

  Since $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, the integration we need to do is:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int \frac{1}{(2\pi\sigma)^d} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z})\|^2 - \frac{1}{2}\|\mathbf{z}\|^2\right) d\mathbf{z} \tag{12}$$

- Nobody knows an analytical solution
  - ... unless $\mathbf{f}$ is linear
- Numerical integration extremely difficult, as always

Estimation by MAP approximation of likelihood

- Since exact likelihood in a DLVM (=generative model) is very difficult to compute, resort to approximations
- The simplest approximation is *Maximum A Posteriori (MAP)*
  - Treat the latent variables like parameters, maximize likelihood with respect to them as well

$$\max_{\boldsymbol{\theta}, \mathbf{z}_1, \ldots, \mathbf{z}_N} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i | \mathbf{z}_i) + \log p(\mathbf{z}_i) \tag{13}$$

  - Often works well, but computation may be hard since so many quantities to optimize: proportional to N which can be millions.

Estimation by Variational Autoencoders (VAE)

- Approximation of likelihood based on variational inference
- "variational" $\approx$ uses an approximative function that is also optimized as part of the approximation
  - Here approximation of conditional pdf $p(\mathbf{z}|\mathbf{x})$ by $q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x})$
  - If approximation by $q_{\boldsymbol{\alpha}}$ is exact for some $\boldsymbol{\alpha}$, method finds that and approximation of likelihood becomes exact too
- Approximator $q_{\boldsymbol{\alpha}}$ should be easy to evaluate *and* easy to sample from (cf. noise in NCE)
- The objective is called ELBO, "evidence lower bound":

$$J(\boldsymbol{\theta}) = \max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \int q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x}(i)) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}(i), \mathbf{z})}{q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x}(i))} d\mathbf{z}$$

  which "internally" maximizea $\boldsymbol{\alpha}$ to make approximation good
- Integration in ELBO is relatively easy by sampling from q !
- A lot of details omitted here...

Model usually used with Variational Autoencoders (VAE)

- Although VAE is a general estimation method, it is usually associated with a particular model (almost defined above)
  1. Latent vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_m)$, i.e. white / independent, as above
  2. Observed data:

     $$\mathbf{x} = \mathbf{f}_\theta(\mathbf{z}) + \mathbf{n} \tag{14}$$

     with $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^d$, where m usually much less than d
- Main point is to reduce dimension: manifold learning
- "Training a VAE" typically means estimating this model using the ELBO
- A lot of confusion in the literature because of this double meaning of "VAE"

VAE as autoencoder

- The estimation crucially includes posterior $p(\mathbf{z}|\mathbf{x})$, approximated by q
- Intuitively plausible:
  to estimate the model, you have to infer the latents
- The $\mathbf{z}$ are then used to generate the data as

$$\hat{\mathbf{x}} = \mathbf{f}_\theta(\mathbf{z}) \tag{15}$$

  and due to Gaussian noise $\mathbf{n}$, the least-squares error $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$ appears in the objective
- This turns out to be very similar to a dimension-reducing autoencoder:
  - (Probabilistic) encoding by $q_\alpha(\mathbf{z}|\mathbf{x})$
  - Decoding by $\mathbf{f}_\theta$
  - ... but more complicated than earlier autoencoders
- It is fashionable to use a VAE for simple dimension reduction, but an ordinary AE would often be a better idea !
  - IMHO: only use VAE if you have a good reason
- (For more on VAEs, see Wikipedia entry, or Prince's book)

Another estimation approach, based on classification

- Consider fundamental problem:
  - We are given two data sets,
    $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$
  - How to measure the similarity of the underlying distributions, $p_\mathbf{x}$ and $p_\mathbf{y}$, *based on those data sets*? (no pdf's given)
  - Related to problem of defining a divergence (see EBM1 slides)
    - but here we have no pdf's, only data:
      methods considered in earlier lectures cannot be used
  - In general, a very difficult problem
- Approach here:
  1. Train a classifier to discriminate between **X** and **Y**
  2. If classification works well (high accuracy)
     $\rightarrow$ the datasets are very different from each other
  3. If classification fails (close to chance level)
     $\rightarrow$ the datasets are similar to each other
- *Distance between distributions* $\sim$ *Classification accuracy*

Generative adversarial networks: basic idea

- A principle of training deep generative networks (DLVM)
- Adjust parameters of the latent variable model so that the distribution of *generated* data is as close as possible to the distribution of the *observed* data
- In more detail, repeat the following:
  1. Generate data from the model with current parameters $\boldsymbol{\theta}$
  2. Train another NN to discriminate between this generated data and the real data
  3. Consider the *negative* of the objective minimized by the classifier (as a simple proxy for classification error) as a measure of distance between distributions
  4. Reduce that distance by gradient descent on $\boldsymbol{\theta}$
- Converges, in theory, when the data generated from model has same distribution as real data
  - That means we fit the model to the data perfectly!
  - We must have found the optimal $\boldsymbol{\theta}$
- This is kind of SSL?

Generative adversarial networks: some typical choices

- While GAN is a general principle for estimating DLVM, usually it is used with a particular model:
  1. Latent vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_m)$, i.e. white / independent, as above
  2. Observed data:
  $$\mathbf{x} = \mathbf{f}_\theta(\mathbf{z}) \tag{16}$$
  with $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^d$
  (here, m can be just about anything, more relaxed than VAE)
- When people say they "train a GAN" it usually means fitting this model with the GAN principle
  - ambiguous, just like "training a VAE" we saw above
- Classifier is pretty much always logistic
- Typically: size of generated dataset = size of real data set
- Optimization by stochastic gradient, but there are some special problems...

Generative adversarial networks: definition

- Denote by $\mathbf{X}$ the whole observed dataset $\mathbf{x}_1, \ldots, \mathbf{x}_N$
- Assume *generative NN*

$$\mathbf{x} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}) \tag{17}$$

  where $\mathbf{z}$ is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\boldsymbol{\theta}$ are NN parameters
- Denote by $\mathbf{Y}(\boldsymbol{\theta})$ a sample of "artificial" data generated using the above Eq. (17) with parameter $\boldsymbol{\theta}$.
- Denote by $J_{\mathrm{lr}}(\mathbf{X}, \mathbf{Y}; \boldsymbol{\alpha})$ the loss (=negative likelihood) of logistic regression, $\boldsymbol{\alpha}$ being parameters of a *discriminating NN*
  - Likelihood (negative loss) is *maximized* to learn classification:

$$\max_{\boldsymbol{\alpha}} -J_{\mathrm{lr}}(\mathbf{X}, \mathbf{Y}(\boldsymbol{\theta}); \boldsymbol{\alpha}) \tag{18}$$

  - (Signs may be confusing: trying to follow literature here)
- The GAN learning principle is then

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha}} -J_{\mathrm{lr}}(\mathbf{X}, \mathbf{Y}(\boldsymbol{\theta}); \boldsymbol{\alpha}) \tag{19}$$

Generative adversarial networks as minimax problem

- Consider the GAN optimization problem we just derived:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha}} -J_{\mathrm{lr}}(\mathbf{X}, \mathbf{Y}(\boldsymbol{\theta}); \boldsymbol{\alpha}) \tag{20}$$

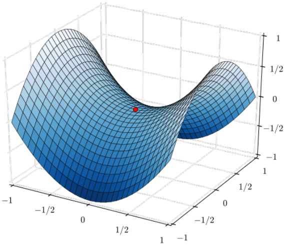
  where J_{lr} is negative likelihood of logistic regression.
- This is *adversarial*: The generating network is trying to choose $\boldsymbol{\theta}$ so that the discriminator NN fails
- *Minimax* problem: interleaved minimization and maximization

→ Tries to find a *saddle point*

- Note: No way you could transform it to "min-min" by changing signs

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha}} -J(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \min_{\boldsymbol{\theta}} - \min_{\boldsymbol{\alpha}} J(\boldsymbol{\theta}, \boldsymbol{\alpha})$$

  since now negative sign appears in the middle; it is not $\min_{\boldsymbol{\theta},\boldsymbol{\alpha}} J(\boldsymbol{\theta}, \boldsymbol{\alpha})$



saddle point

Generative adversarial networks: optimization

- How to actually solve the optimization of:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha}} -J_{\mathrm{lr}}(\mathbf{X}, \mathbf{Y}(\boldsymbol{\theta}); \boldsymbol{\alpha}) \tag{21}$$

- In principle, optimize $\max_{\boldsymbol{\alpha}} -J_{\mathrm{lr}}(\mathbf{X}, \mathbf{Y}(\boldsymbol{\theta}); \boldsymbol{\alpha})$ until convergence, and then take gradient step with respect to $\boldsymbol{\theta}$.
- This leads to another interpretation: $\max_{\boldsymbol{\alpha}} -J_{\mathrm{lr}}(\mathbf{X}, \mathbf{Y}(\boldsymbol{\theta}); \boldsymbol{\alpha})$ is objective being minimized, but it is *changing* (learned?)
- In practice, could just do stochastic gradient method with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ in alternation

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \mu_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\alpha}} J_{\mathrm{lr}}(\mathbf{X}, \mathbf{Y}(\boldsymbol{\theta}); \boldsymbol{\alpha}) \tag{22}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mu_{\boldsymbol{\alpha}} \nabla_{\boldsymbol{\theta}} J_{\mathrm{lr}}(\mathbf{X}, \mathbf{Y}(\boldsymbol{\theta}); \boldsymbol{\alpha}) \tag{23}$$

  for *minibatches* $\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}(\boldsymbol{\theta})$, and two step sizes $\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\alpha}}$.
- Very challenging in practice! Beware of instability!

Generative adversarial networks: applications

- Overwhelmingly, in data generation (image, video)

(Karras et al, 2018)

- GAN's were the beginning of the Generative AI boom
- Now, partly superseded by generative diffusion models
  - In diffusion models, standard optimization problem: not adversarial

Reminder: Unsupervised learning can have different goals

Reminder: Unsupervised learning can have different goals

1) Accurate model of data distribution?
   - E.g. Energy-based models *(Done!)*

Reminder: Unsupervised learning can have different goals

1) Accurate model of data distribution?
   - E.g. Energy-based models *(Done!)*
2) Sampling points from data distribution?
   - E.g. Generative Adversarial Networks, MCMC, diffusion models *(Done!)*

Reminder: Unsupervised learning can have different goals

1) Accurate model of data distribution?
   - E.g. Energy-based models *(Done!)*
2) Sampling points from data distribution?
   - E.g. Generative Adversarial Networks, MCMC, diffusion models *(Done!)*
3) Useful features for supervised learning? *(Done!)*
   - Contrastive learning, autoencoders, many kinds of SSL

Reminder: Unsupervised learning can have different goals

1) Accurate model of data distribution?
   - E.g. Energy-based models *(Done!)*
2) Sampling points from data distribution?
   - E.g. Generative Adversarial Networks, MCMC, diffusion models *(Done!)*
3) Useful features for supervised learning? *(Done!)*
   - Contrastive learning, autoencoders, many kinds of SSL
4) Reveal identifiable structure in data, "disentangle" latent quantities?
   - Independent Component Analysis *(next)*

Problem of identifiability

- Do we actually estimate the model properly?
- Do we actually get the "correct" $\mathbf{f}$ in the model

$$\mathbf{x} = \mathbf{f}_{\theta}(\mathbf{z}) \tag{24}$$

- Definition: a model is *identifiable* if for two different values of $\boldsymbol{\theta}$, the distributions of the data $p_{\mathbf{x}}$ are different
- Identifiability means it is *possible* to find the underlying $\boldsymbol{\theta}$ from the data $\mathbf{x}$, assuming $\mathbf{x}$ is generated from the model
  - In practice, it means there is some *hope* of a unique solution
  - ... but a DL algorithm can give different solutions even for an identifiable model
    - Local minima, random initial points
- For data generation, we may not care about identifiability (?)
- But for understanding data, identifiability is paramount

Unidentifiability of Gaussian latent variables

- Consider the original Gaussian, white latent vector $\mathbf{z}$
- We have for any *orthogonal* transformation $\mathbf{z}' = \mathbf{U}\mathbf{z}$:

$$\mathrm{cov}(\mathbf{z}') = \mathbf{U}\mathrm{cov}(\mathbf{z})\mathbf{U}^T = \mathbf{U}\mathbf{I}\mathbf{U}^T = \mathbf{I} \tag{25}$$

- The covariance stays the same! (for white $\mathbf{z}$)
- But a zero-mean gaussian pdf depends only on the covariance

$\Rightarrow$ $\mathbf{z}$ and $\mathbf{z}'$ have same distribution

- Define $\boldsymbol{\theta}'$ that gives a neural network such that

$$\mathbf{f}_{\boldsymbol{\theta}'}(\zeta) = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{U}^T\zeta) \tag{26}$$

- Then we have:

$$\mathbf{x}' = \mathbf{f}_{\boldsymbol{\theta}'}(\mathbf{z}') \tag{27}$$

- Data is same $\mathbf{x}$ in both cases, but $\boldsymbol{\theta} \neq \boldsymbol{\theta}'$
  (and $\mathbf{z}$ and $\mathbf{z}'$ follow the same prior)

$\Rightarrow$ $\boldsymbol{\theta}$ is not identifiable (and neither is $\mathbf{z}$)

Practical implication of unidentifiability

- Suppose you train a VAE (in the usual sense) on, say, some scientific data
- Crucially: it makes no sense to interpret the individual components as corresponding to some scientific phenomena
  - We already saw this with dimension-reducing autoencoders
- However, this is very often done in the literature, due to general ignorance of this problem
- Same thing applies to a GAN (but such interpretation is rare with GAN)

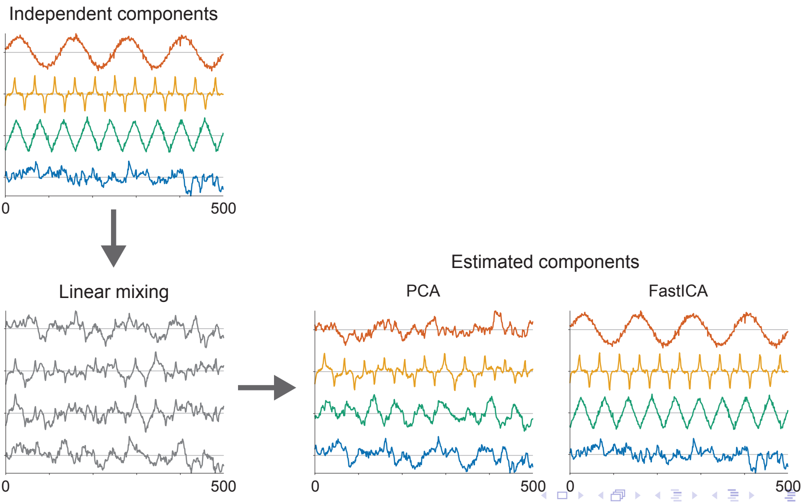
ICA as an identifiable linear generative model

- Linear independent component analysis (ICA)

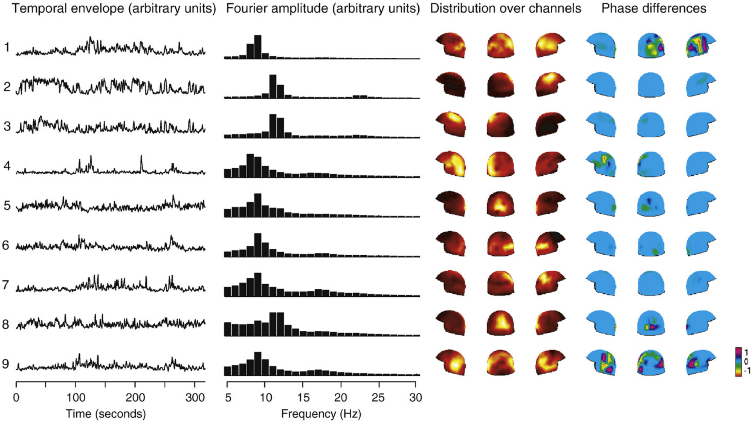
$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{28}$$

  - $\mathbf{x} \in \mathbb{R}^d$ is the observed data
  - $\mathbf{A}$ matrix of constant parameters describing "mixing"
  - Assuming independent, non-Gaussian latent variables s_j, $j = 1, \ldots, d$ (in basic case, dimensions are equal for $\mathbf{s}$ and $\mathbf{x}$)
- ICA is identifiable:
  - Observing only $\mathbf{x}$ we can recover both $\mathbf{A}$ and $\mathbf{s}$
- Fundamental point: non-Gaussianity of the latents enables identifiability
  - together with independence
  - But this works only in linear case...

Linear ICA can separate "sources" from mixtures



Real example of ICA: Brain source separation



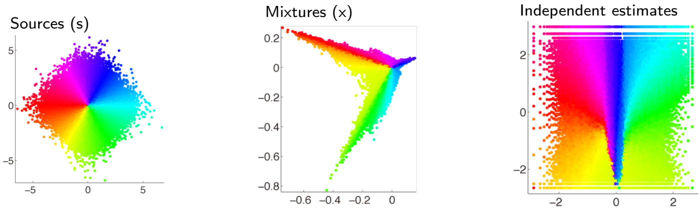
(Hyvärinen, Ramkumar, Parkkonen, Hari, 2010)

How to formulate ICA in the nonlinear case?

- Can we extend ICA to nonlinear case, get identifiable DLVM?
- A simple extension of nonlinear ICA is *not* identifiable
- If we define nonlinear ICA model for random variables x_i as

$$\mathbf{x} = \mathbf{f}(\mathbf{z}) \tag{29}$$

we cannot recover original sources, even if z_i are non-Gaussian.



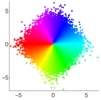
Darmois construction to prove non-identifiability

- For any x_1, x_2, can always construct $y = g(x_1, x_2)$ independent of x_1 as

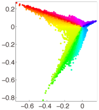
$$g(\xi_1, \xi_2) = P(x_2 < \xi_2 | x_1 = \xi_1) \tag{30}$$

- After obtaining such independent variables we can find a scalar function $h(x_1), h(x_2)$ that transform to any distribution we want
- With nonlinear transformations, non-Gaussianity is meaningless since it can be changed arbitrarily
- Using this construction, we could even take x_1 as independent component which is absurd

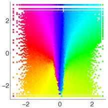
Sources (s)



Mixtures (x)

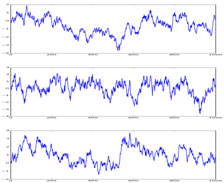


Independent estimates

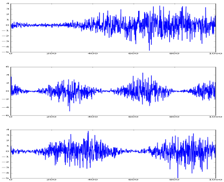


Temporal structure helps in nonlinear ICA

- Theory above considered i.i.d. sampled random variables
- What if we have time series? with specific temporal structure?



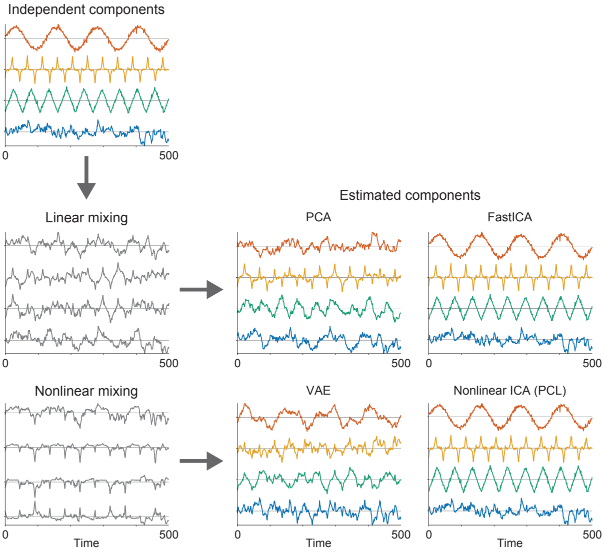
Autocorrelations



Nonstationarity

- Identifiability of nonlinear ICA can be proven
  Can find original sources!

Nonlinear ICA can separate “sources” from mixtures



One identifiable source model: Temporal dependencies

- Assume mixing model $\mathbf{x_t} = \mathbf{f}(\mathbf{s_t})$ where
  - $\mathbf{x}_t$ observed n-dimensional time series
  - $\mathbf{s}_t$ latent n-dimensional independent time series
  - $\mathbf{f}$ invertible (bijective) mixing
- Assume s_t^i temporally dependent and non-Gaussian (and some technical constraints)
- E.g., non-Gaussian AR model with non-quadratic G:

$$\log p(s_t^i | s_{t-1}^i) = G(s_t^i - \rho s_{t-1}^i)$$

One identifiable source model: Temporal dependencies

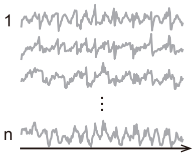
- Assume mixing model $\mathbf{x_t} = \mathbf{f}(\mathbf{s_t})$ where
  - $\mathbf{x}_t$ observed n-dimensional time series
  - $\mathbf{s}_t$ latent n-dimensional independent time series
  - $\mathbf{f}$ invertible (bijective) mixing
- Assume s_t^i temporally dependent and non-Gaussian (and some technical constraints)
- E.g., non-Gaussian AR model with non-quadratic G:

$$\log p(s_t^i | s_{t-1}^i) = G(s_t^i - \rho s_{t-1}^i)$$

- This is Identifiable!
- Why would this work? Impose independence over time lags $\rightarrow$ more constraints $\rightarrow$ unique solution
- Estimation by VAE, GAN, even MLE possible; also more heuristic SSL methods (next)

Permutation-contrastive learning

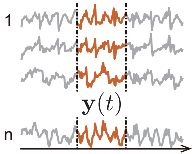
- SSL on temporal dependencies



Permutation-contrastive learning

- SSL on temporal dependencies
- Take short time windows as new data

$$\mathbf{y}(t) = \big(\mathbf{x}(t), \mathbf{x}(t-1)\big)$$



Permutation-contrastive learning

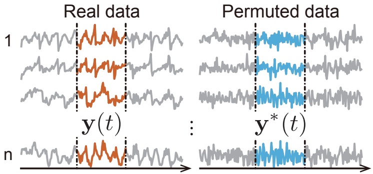
- SSL on temporal dependencies
- Take short time windows as new data

$$\mathbf{y}(t) = \big(\mathbf{x}(t), \mathbf{x}(t-1)\big)$$

- Create randomly time-permuted data

$$\mathbf{y}^*(t) = \big(\mathbf{x}(t), \mathbf{x}(t^*)\big)$$

with t^* a random time point.



Permutation-contrastive learning

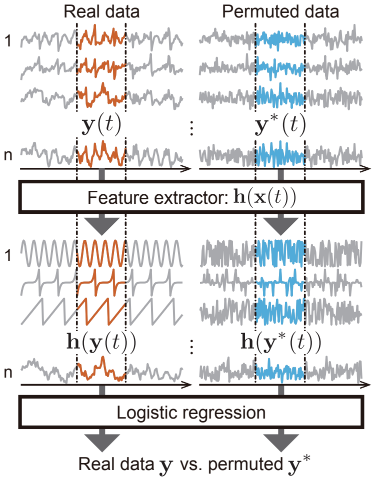
- SSL on temporal dependencies
- Take short time windows as new data

$$\mathbf{y}(t) = (\mathbf{x}(t), \mathbf{x}(t-1))$$

- Create randomly time-permuted data

$$\mathbf{y}^*(t) = (\mathbf{x}(t), \mathbf{x}(t^*))$$

  with t^* a random time point.
- Train NN to discriminate $\mathbf{y}$ from $\mathbf{y}^*$
- Performs Nonlinear ICA for temporally dependent components! Quite surprising...



Conclusion on generative models

- Deep Latent Variable Models: one approach to unsupervised modelling of data
- Alternative to energy-based models
- Estimation of DLVM is challenging
  - EBMs may be better from that viewpoint
- Strength: DLVM automatically generates new data points
  - GANs were original success in GenAI
- DL often uses trivial models for latents: Gaussian and white

$\Rightarrow$ Serious problems of identifiability

- Nonlinear ICA solves identifiability
  - More difficult than linear ICA where non-Gaussianity is enough
  - We need e.g. nonstationarity or temporal dependencies