Dimensionality Reduction

ioan.moldovan@tora.com

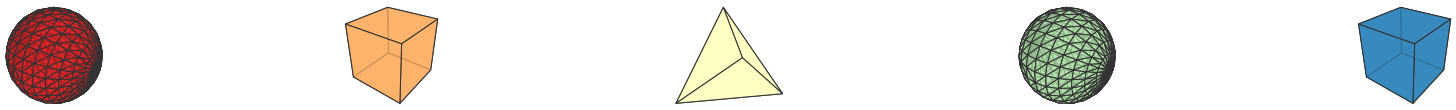
PyData Cluj-Napoca meetup 3, 2019.05.07

PDF slides are incomplete (no animations, no footnotes, no presenter notes, partial info on slides).

See slides.html?presentme=true for a nicer version.

What is dimensionality reduction

Retaining relevant data from a [large] set of features,



projecting to a lower dimensional feature subspace without losing important information

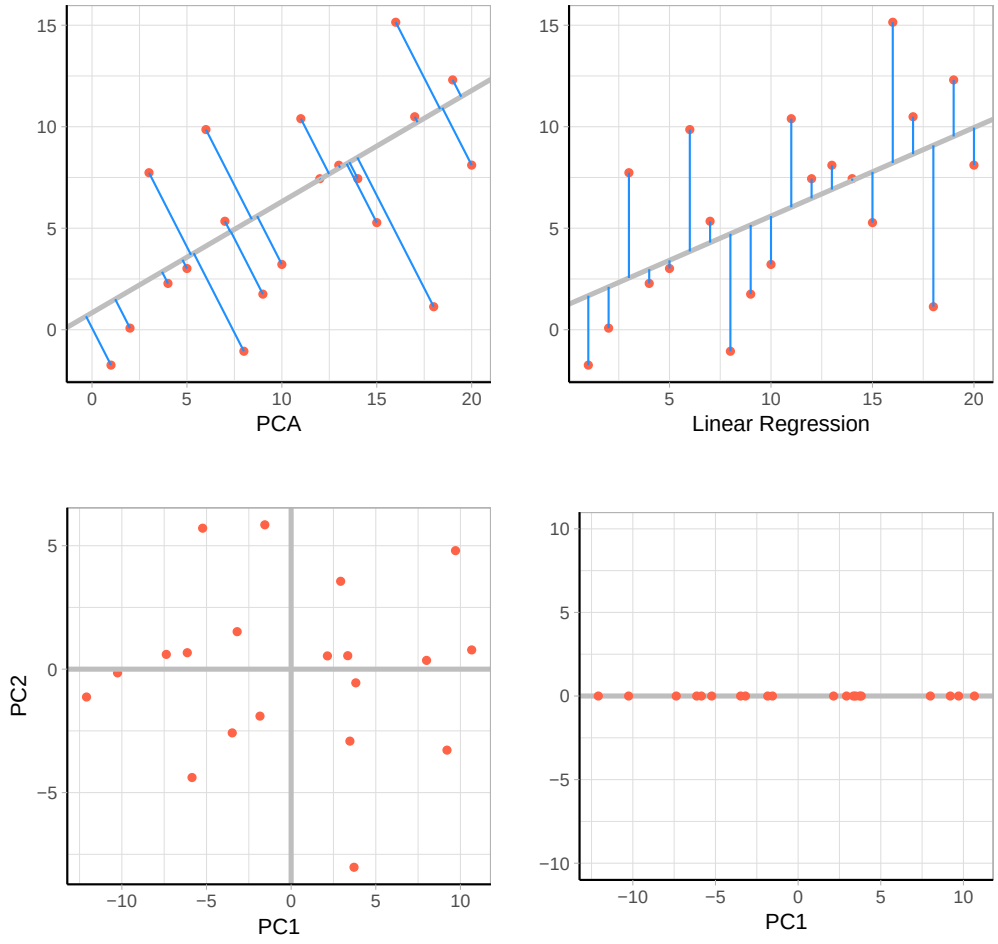
depending on the problem to be solved.

Summary

- brief look at PCA (to know what we are talking about)
- why dimensionality reduction
- feature selection
- supervised feature extraction: LDA, QDA
- unsupervised feature extraction: PCA, sparse PCA, kernel PCA, ICA
- non-linear dimensionality reduction: MDS, IsoMap, LLE, LLSTA, SOM
- [mainly] visualization: t-SNE, UMAP
- autoencoders: deep, sparse, stacked, contractive, denoising, recurrent, VAE, beta-VAE, UAE
- adversarial networks: cVAEGAN

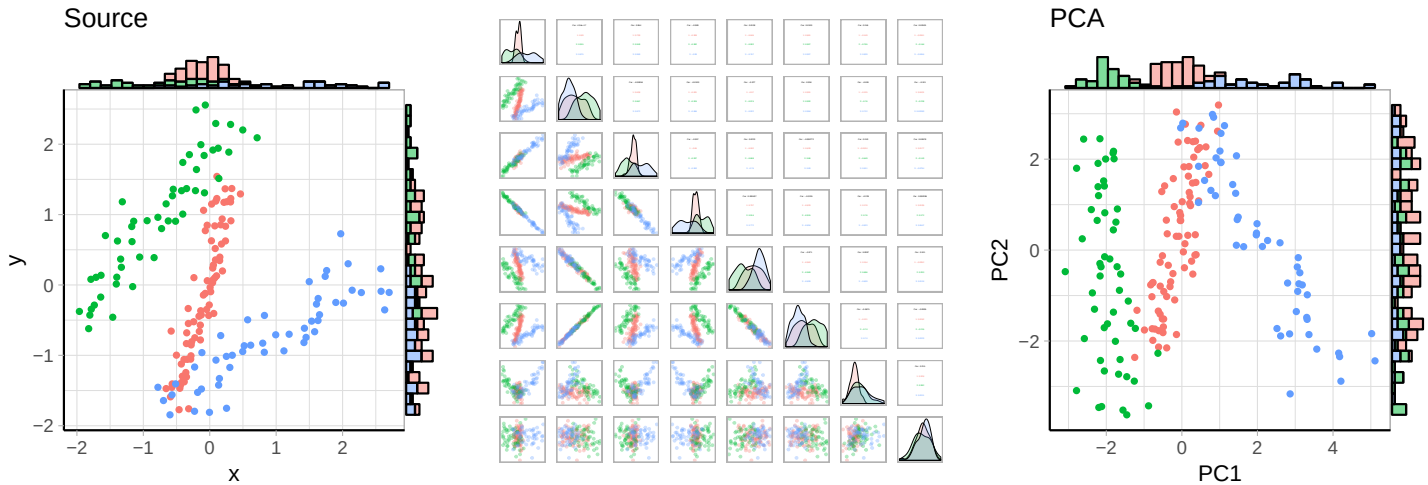
PCA

- finds a linear transformation of the input data such that the new frame of reference "better explains the data"



Principal Component Analysis (PCA)

- generated an 8-dimensional dataset from the points on the left plot
- PCA is able to reconstruct a rotation of the source signal in the first 2 components



Why dimensionality reduction

When:

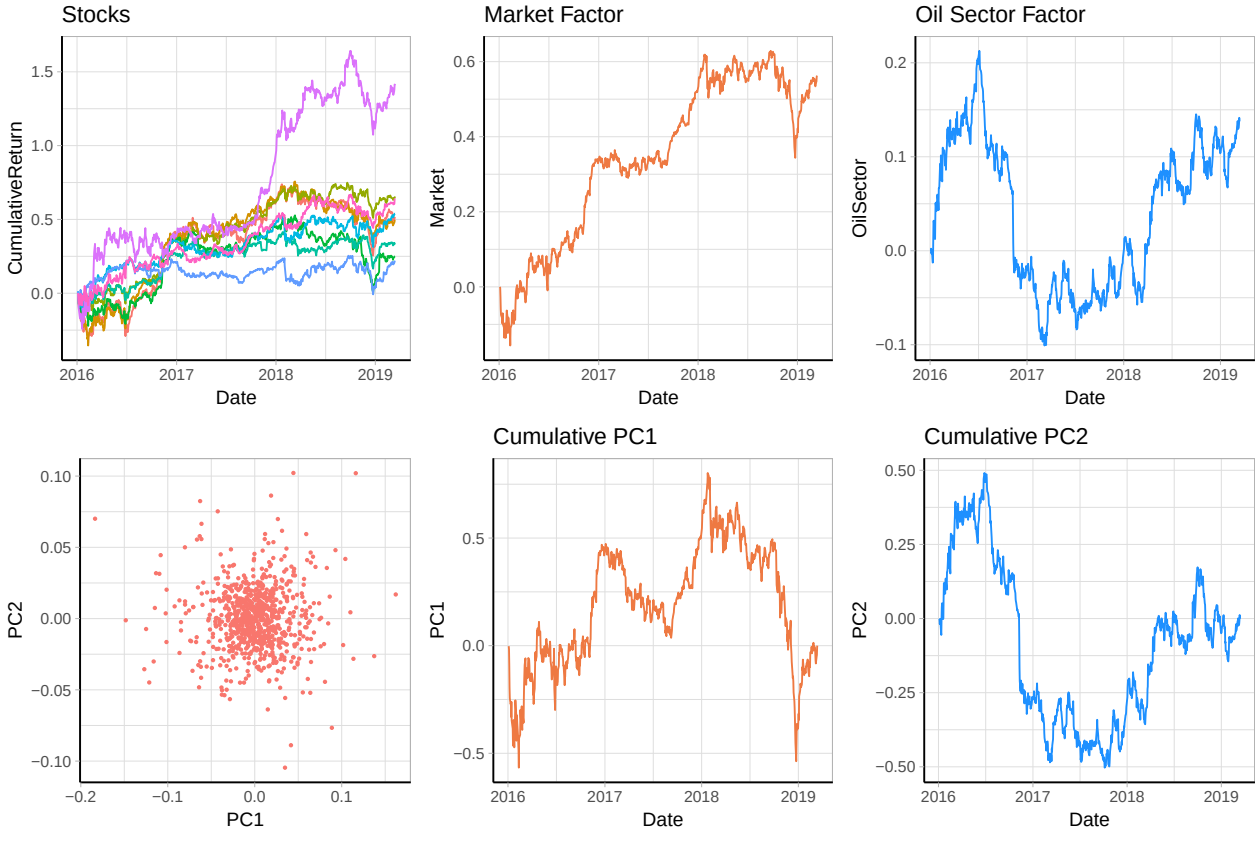
- lots of dimensions
- multiple aligned time series with common underlying factors

Why:

- exploratory data analysis
- visualizations
- extraction of underlying (or relevant) factors
- better data separability, informative clustering
- robust learning, less overfitting, improved supervised or unsupervised models performance
- noise reduction
- anomaly detection
- data compression
- less computational resources

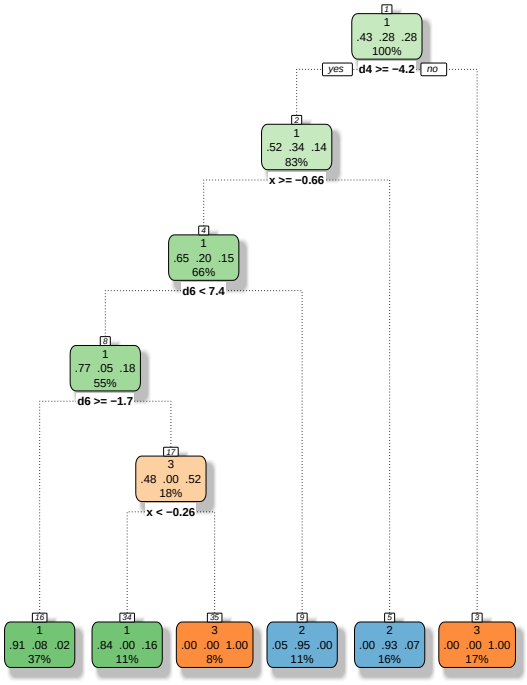
Why dimensionality reduction

- extraction of underlying factors: various "input features" are just measurements which can reflect the same underlying (maybe hidden) factors. Sometimes we can extracts those factors or get better combined measurements

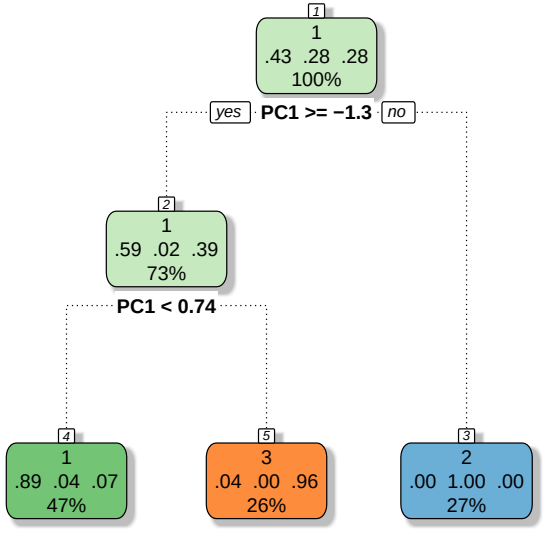


Why dimensionality reduction

- sample Decision Tree on the 8-dim dataset used for the PCA example before, trained on 8 raw features and on 2 PCA components



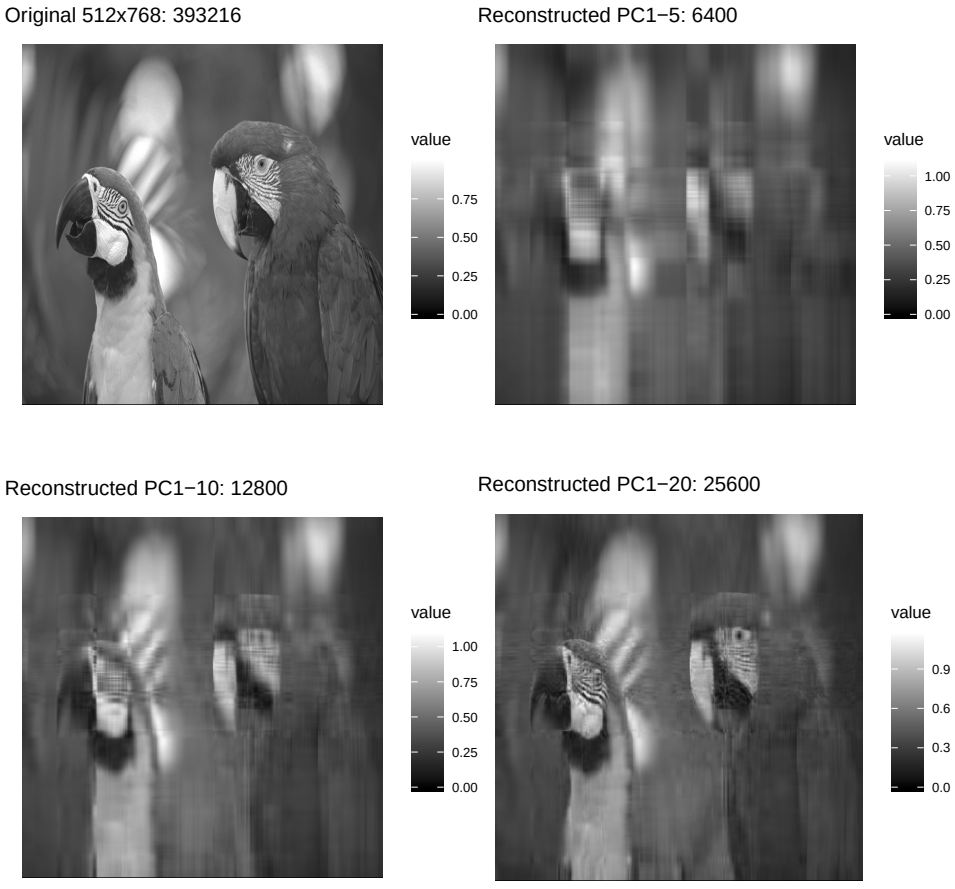
Raw data, accuracy 0.93



Transformed data, accuracy 0.94

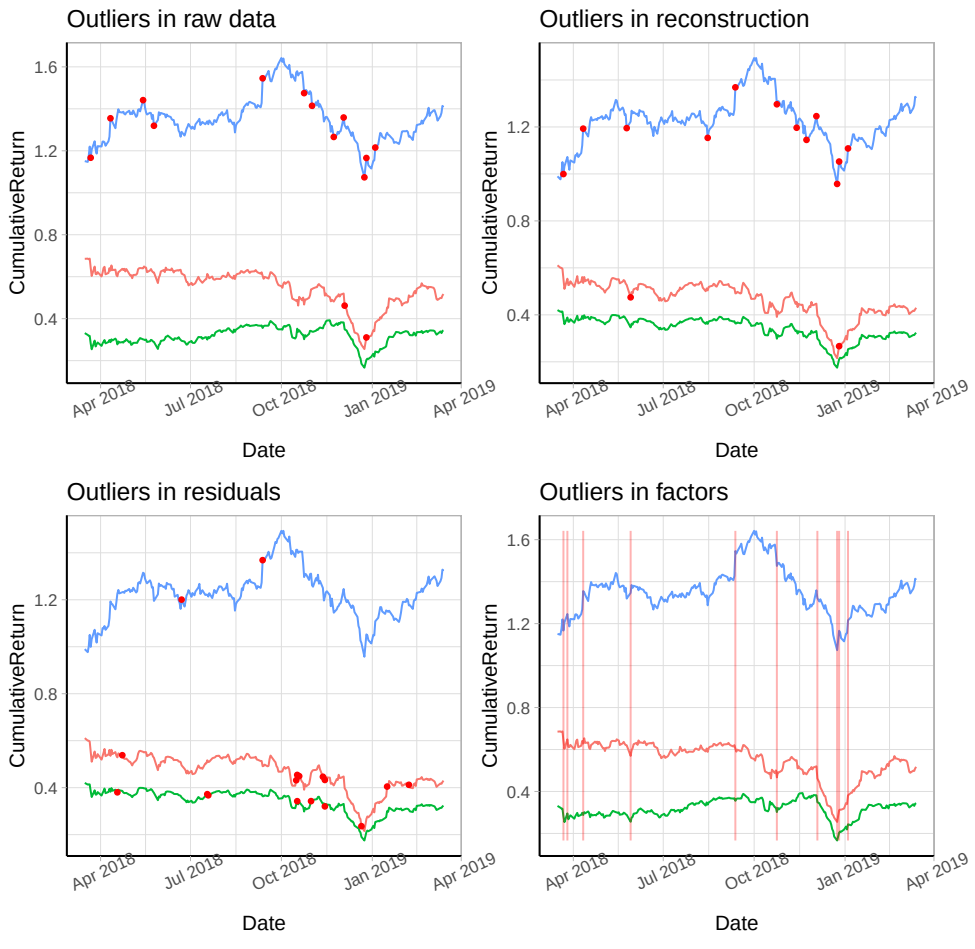
Why dimensionality reduction

- data compression
- why it works: successive data samples have some relation one to another



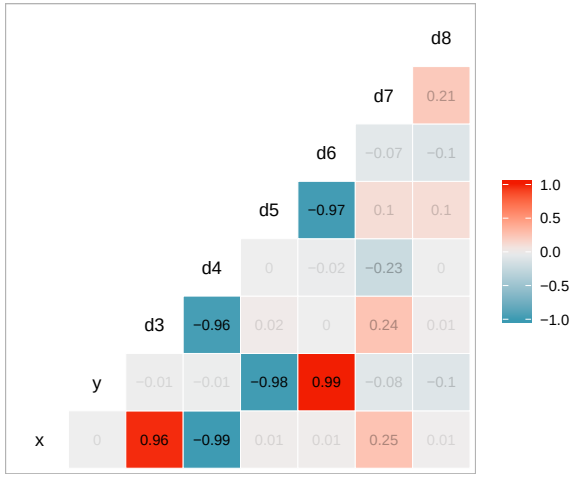
Why dimensionality reduction

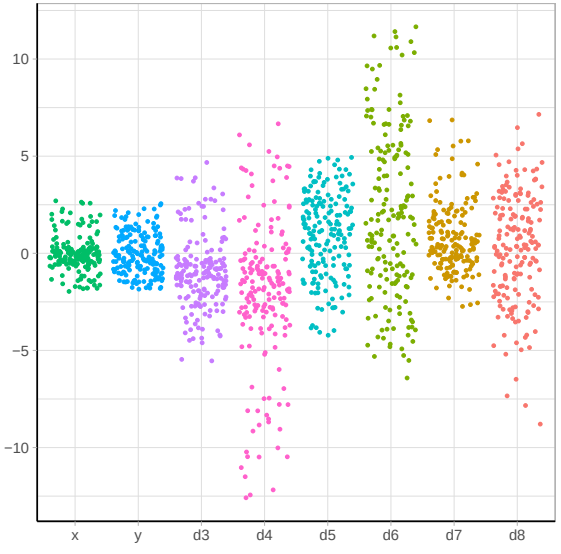
- noise removal and improved anomaly detection



Feature selection

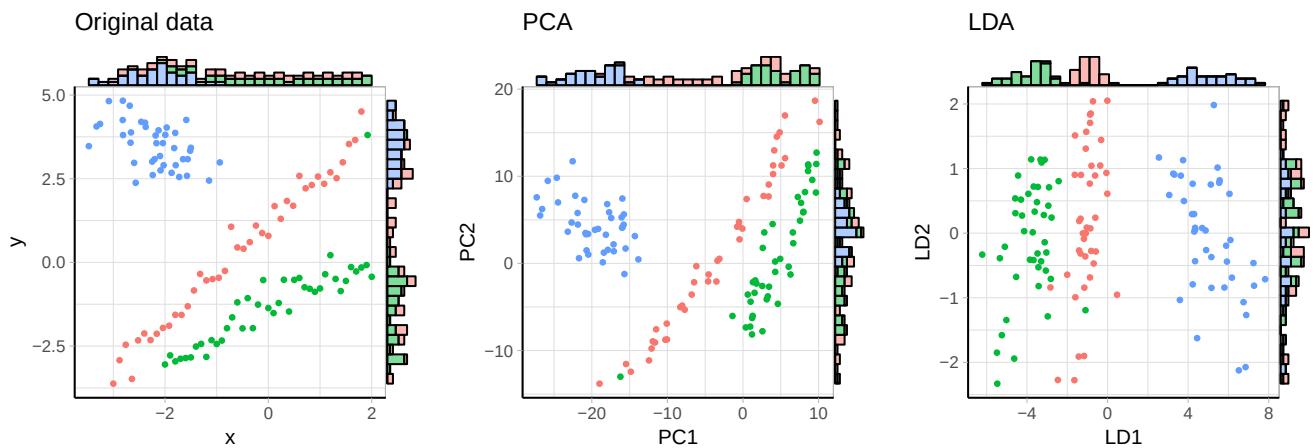
- feature selection vs feature extraction: dropping features vs creating new ones
- low variance filter
- correlation based filtering
- univariate feature selection
- recursive feature elimination
- variable importance based filtering





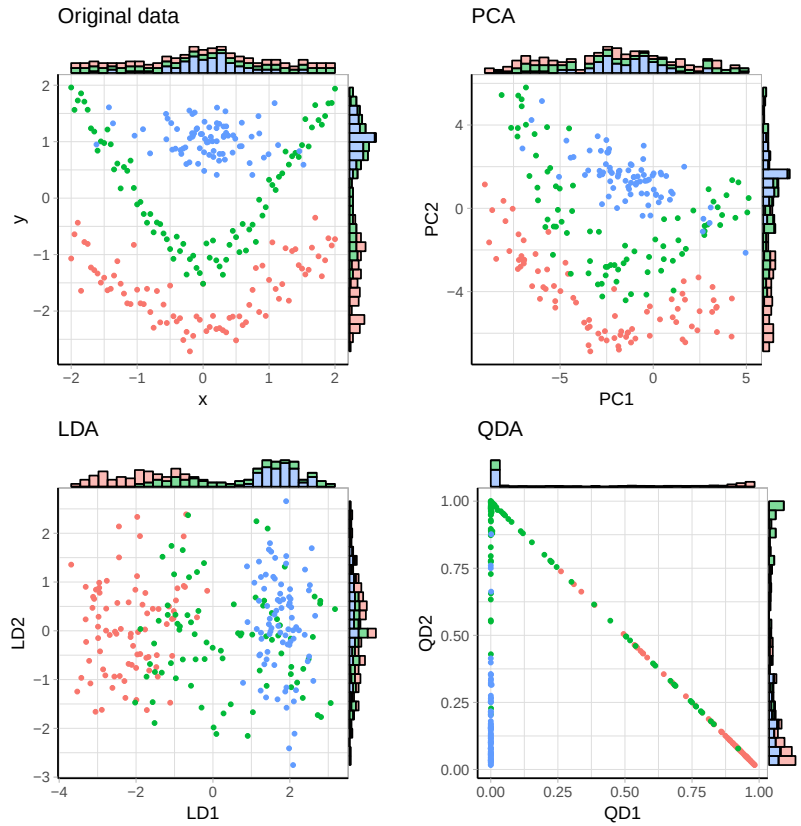
Linear Discriminant Analysis (LDA)

- supervised methods try to reduces the number of dimensions by trying to maximize the class separation
- LDA also computes a decision boundary, it can be used as a classifier
- projections to the first component have the highest class separation (LDA decomposition focus is separation, while PCA decomposition focus is correlation)



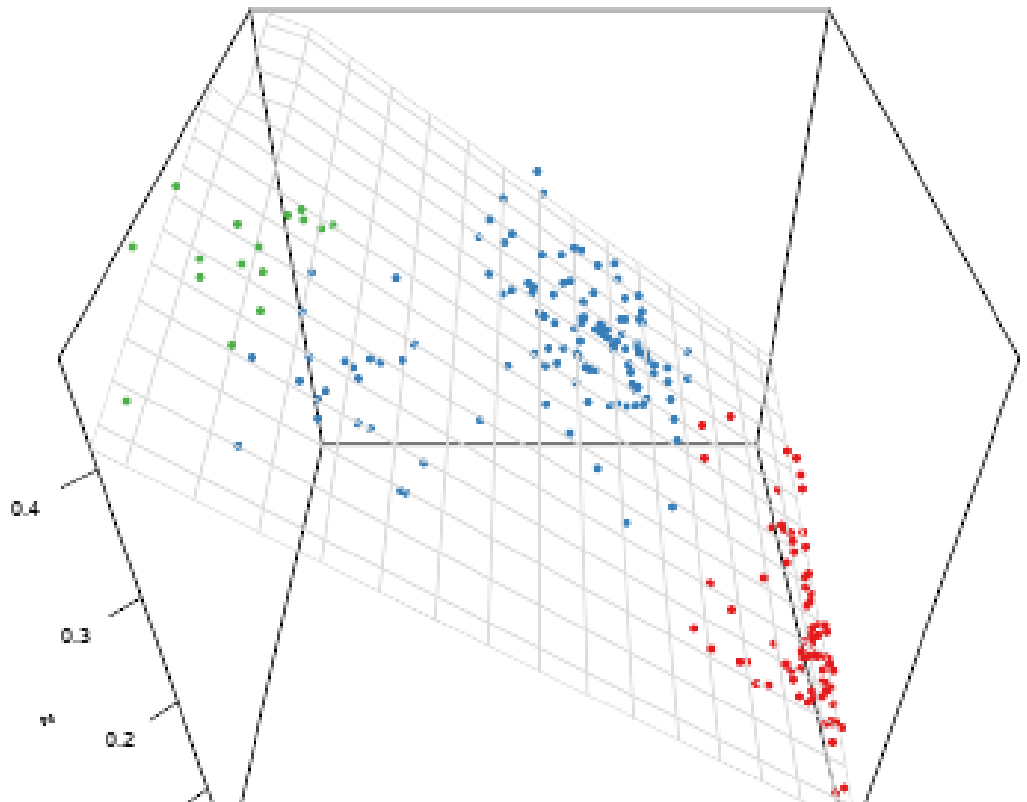
Quadratic Discriminant Analysis (QDA)

- can be used as a classifier
- suitable for non-linearly separable data
- less assumptions about the distributions than LDA, QDA computes both means and covariance for each group



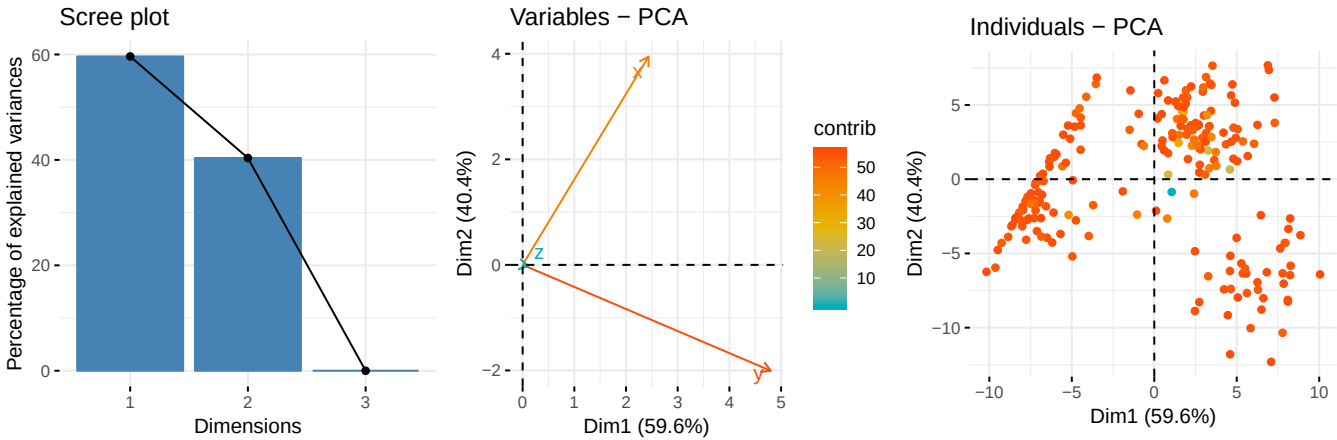
PCA

- when dropping components: projection to lower dimensional space
- the new axes define a lower dimensional subspace, the compressed dataset is built from the new coordinates of the projected points in this subspace



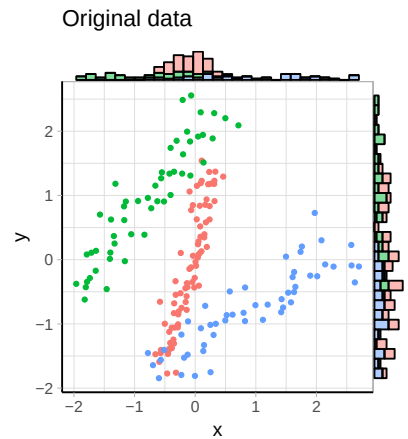
PCA

- what to look at: the principal components, explained variance by axis, direction of initial vectors in the new subspace, datapoints projection, reconstruction (additive from contributions on each axis), estimated dimensionality

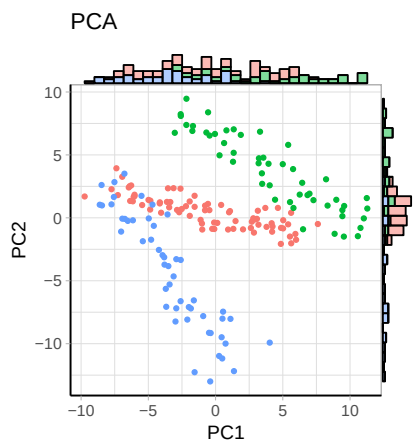


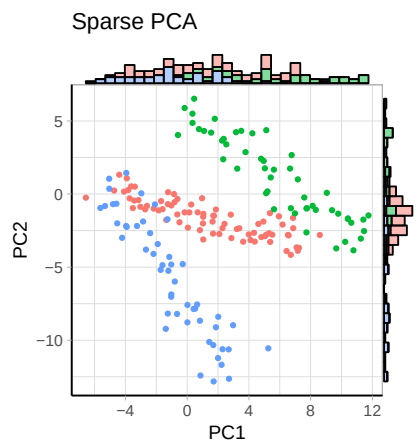
sparse PCA

- PCA vs sparse PCA: both are linear transformations, sparse PCA just another rotation
- iterative algorithm using shrinkage methods (lasso, ridge, elastic net)
- adds a regularization term to the loss function, penalizing non-zero coefficients



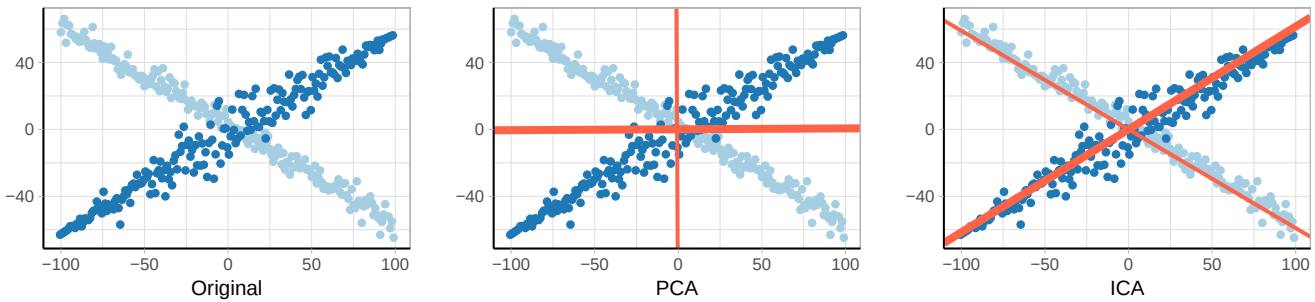
	PCA.PC1	PCA.PC2	SPCA.PC1	SPCA.PC2
x	0.006438	−0.214385	0.000000	0.000000
y	0.214747	0.008046	0.000000	0.000000
d3	0.008414	−0.422412	0.000000	0.000000
d4	−0.035330	0.872124	0.000000	0.999377
d5	−0.430737	−0.022266	0.000000	0.000000
d6	0.870862	0.021968	0.999003	0.000000
d7	−0.032349	−0.116875	0.000000	−0.035287
d8	−0.086850	−0.017457	−0.044634	0.000000





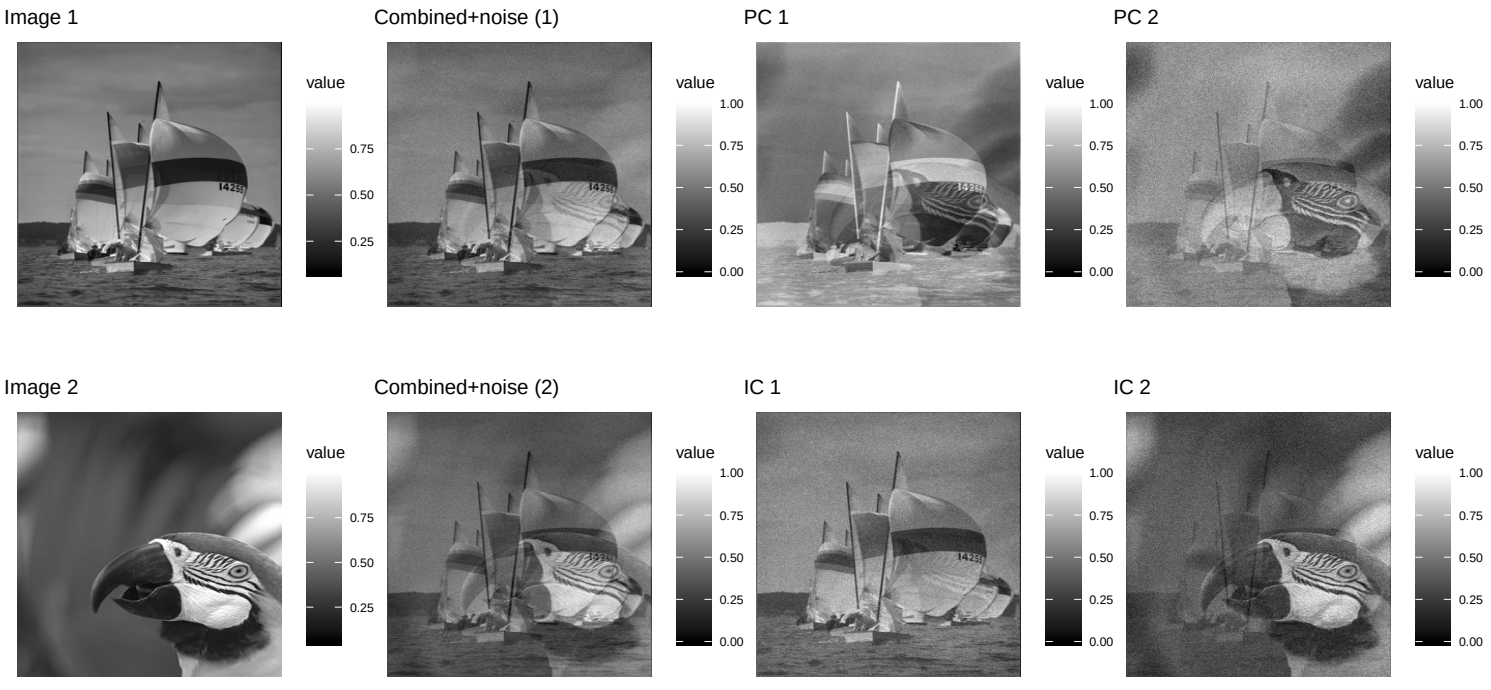
Independent Component Analysis (ICA)

- ICA vs PCA - variance explaining and orthogonal features vs signal explaining and independent features
- iterative algorithms doing minimization of mutual information or maximization of non-gaussianity (negentropy or kurtosis)



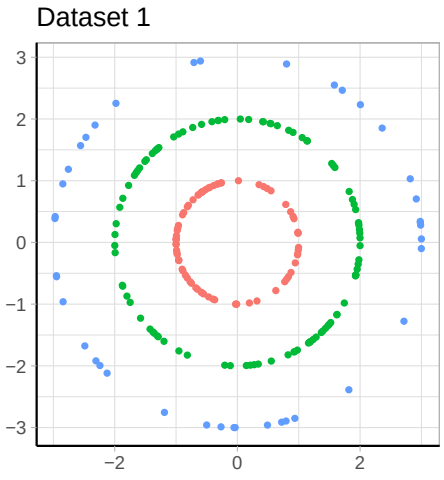
ICA

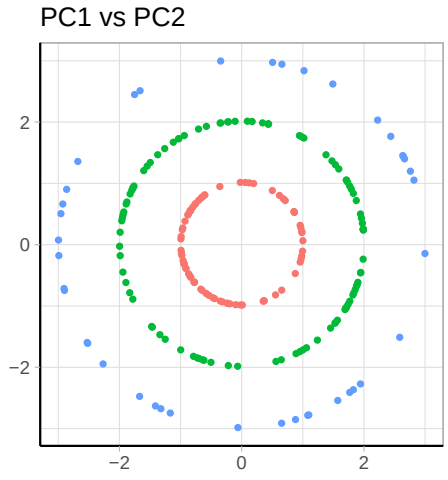
- ICA extracts “independent features”, theoretically the reconstruction of original images should be better if they have low mutual information

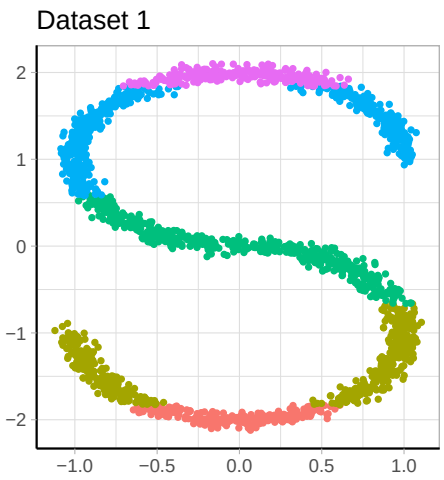


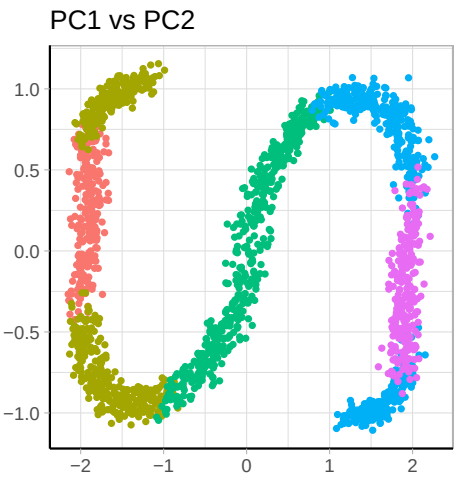
PCA on non-linear data

- sometimes linear transformations are not enoguh



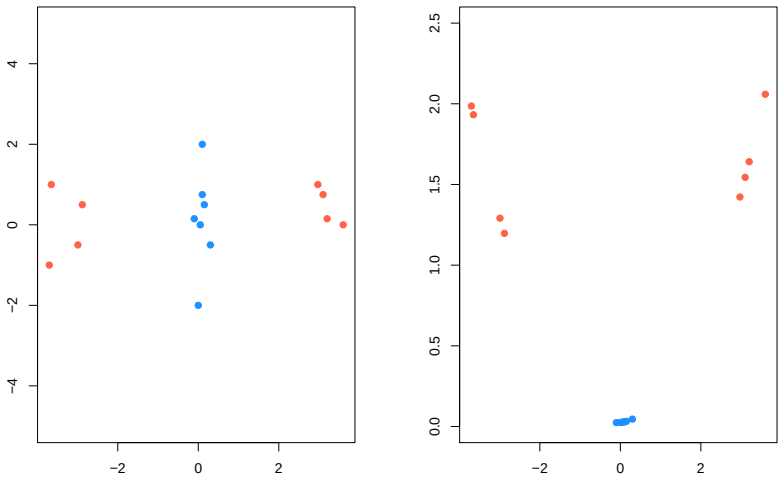






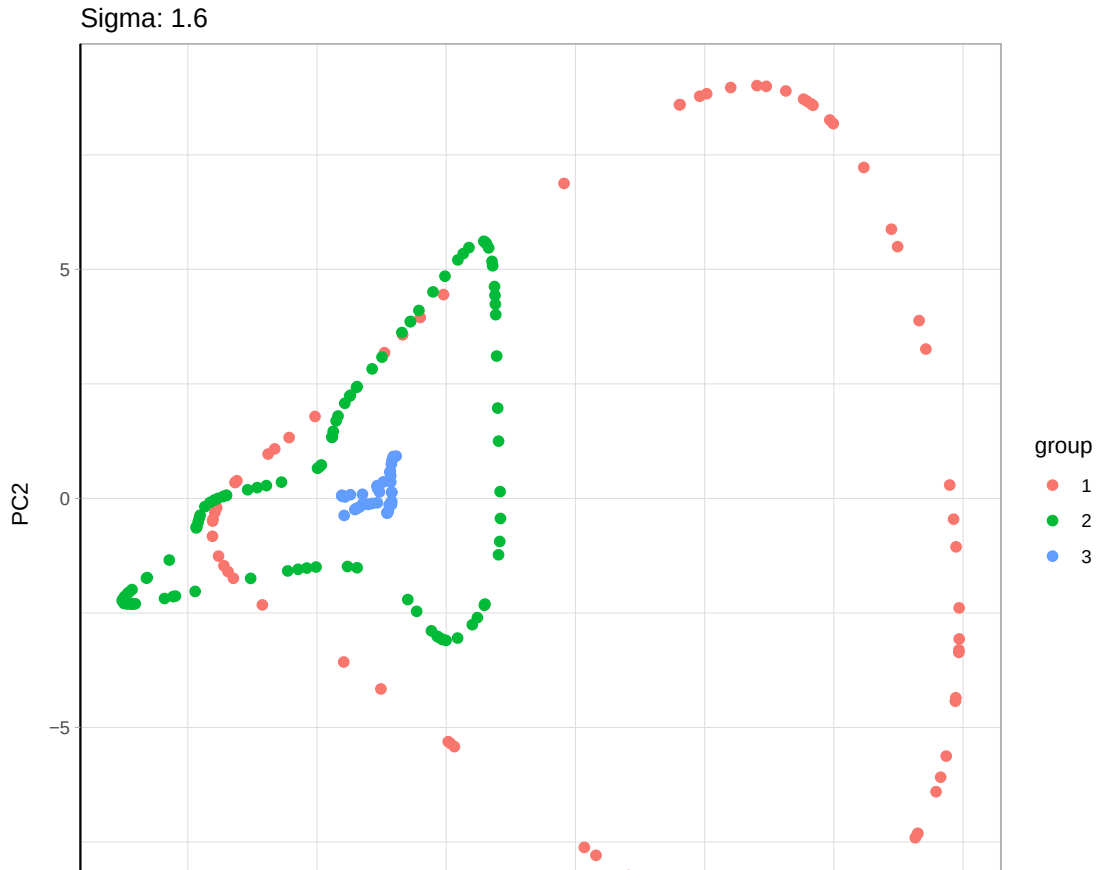
kernel PCA

- "kernel trick": a mapping function computing the similarities between points in a higher dimension in order to obtain better data separability, without actually knowing where the points are in the higher dimensional space
- linear, polynomial, spline, radial basis function (RBF), ANOVA RBF etc.
- a RBF kernel projects into an infinite dimensional space, it is way more complex than the 2D gaussian function on the right



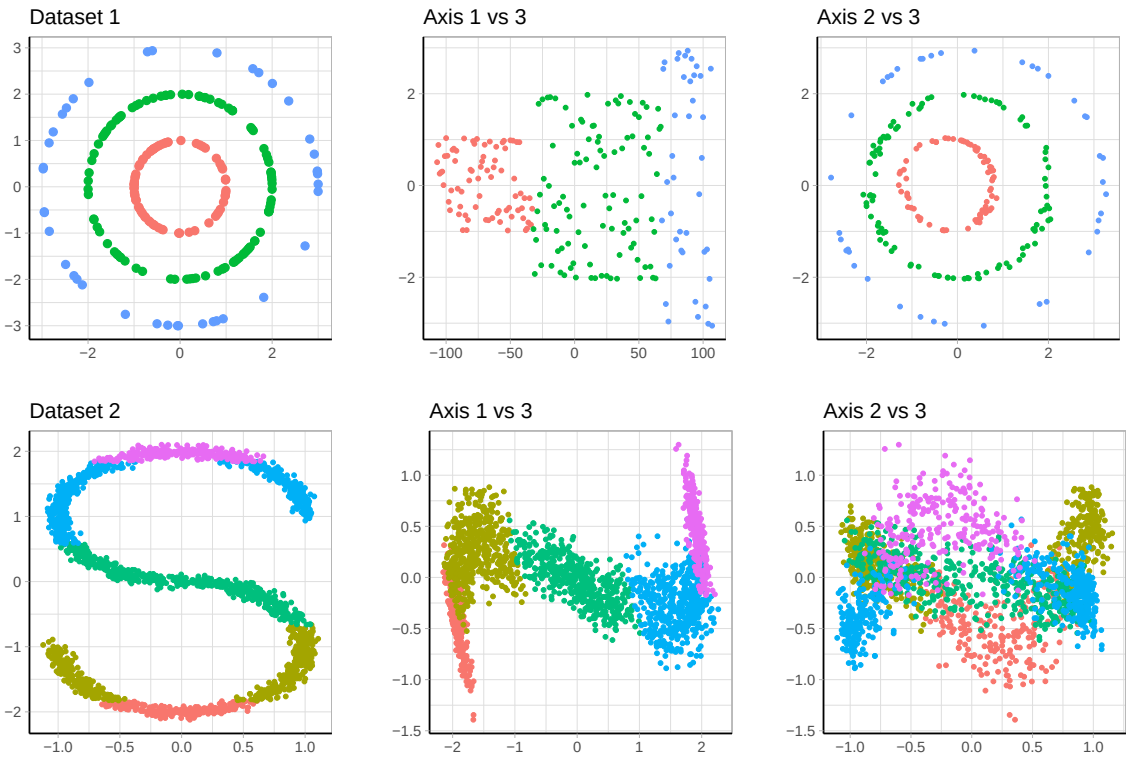
kernel PCA

- maps points without making a projection, it just computes a dot-product between points “as if”" they were from a higher dimension
- kernel parameters tuning: various methods, usually grid search
- models incorporate external knowledge when applying a certain kernel or technique (including by the choice of the parameters)



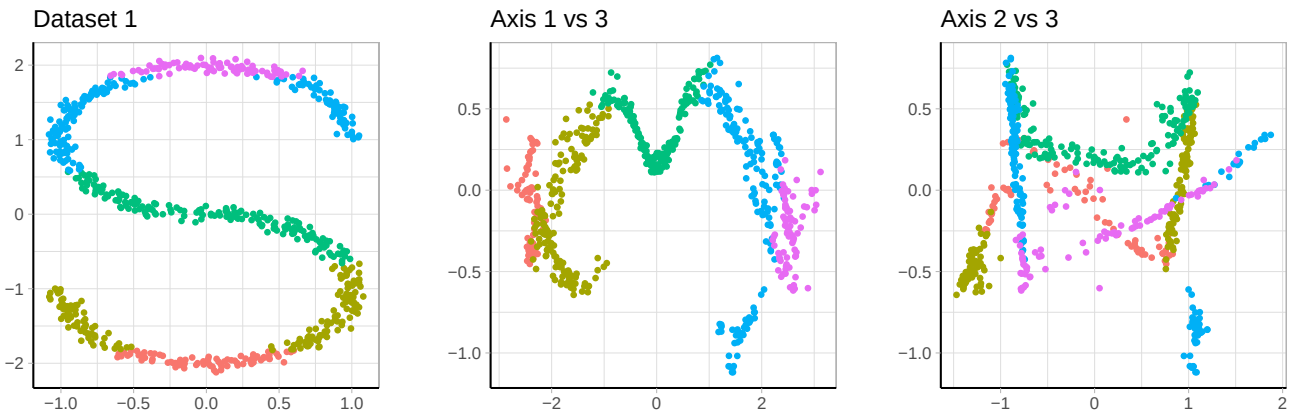
Multidimensional Scaling (MDS)

- non-linear transformation, applied on distances (can be non-euclidian)
- transforms the proximity metric to fitted distances in a lower dimensional space, such that the disparities are mapped as good as possible
- preserves topology



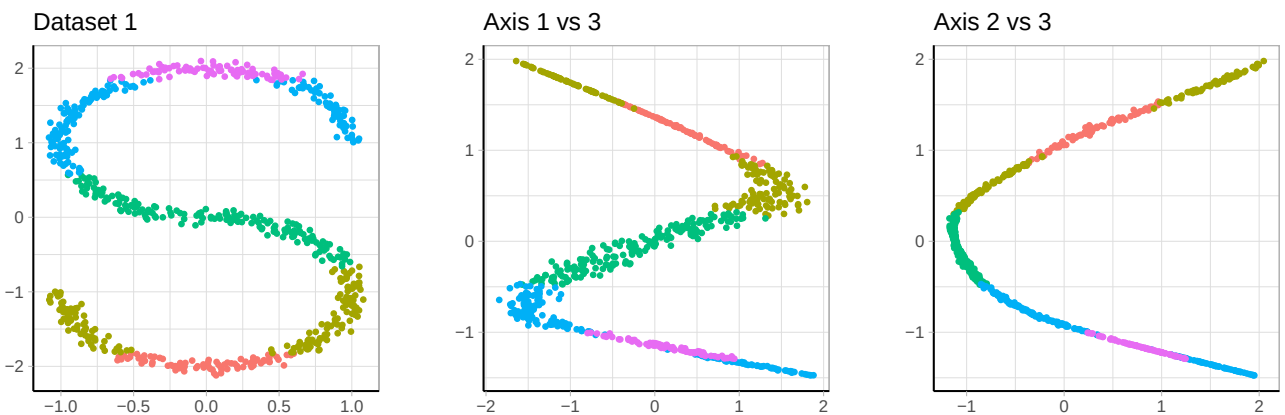
IsoMap

- Isometric Feature Mapping Ordinationhttps://web.mit.edu/cocosci/Papers/sci_reprint.pdf
- uses "geodesic" distances: retains only some of the graph distances among objects (the smaller ones) and estimates all dissimilarities as shortest path distances



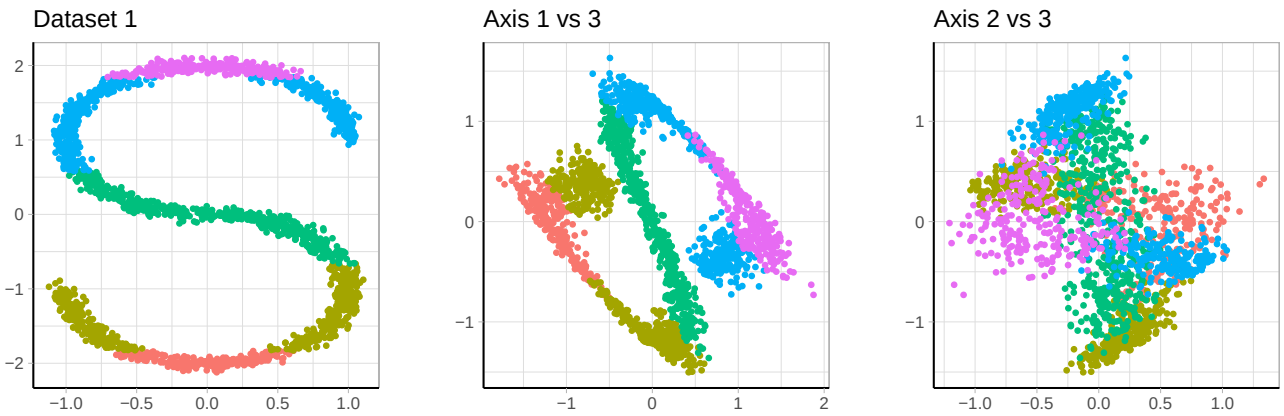
Locally Linear Embedding (LLE)

- finds neighbours, computes the weights that best reconstruct the points from its neighbors as linear combinations, map to embedded coordinates by computing the best reconstruction from those weights
- “think globally, fit locally”http: //www.jmlr.org/papers/volume4/saul03a/saul03a.pdf



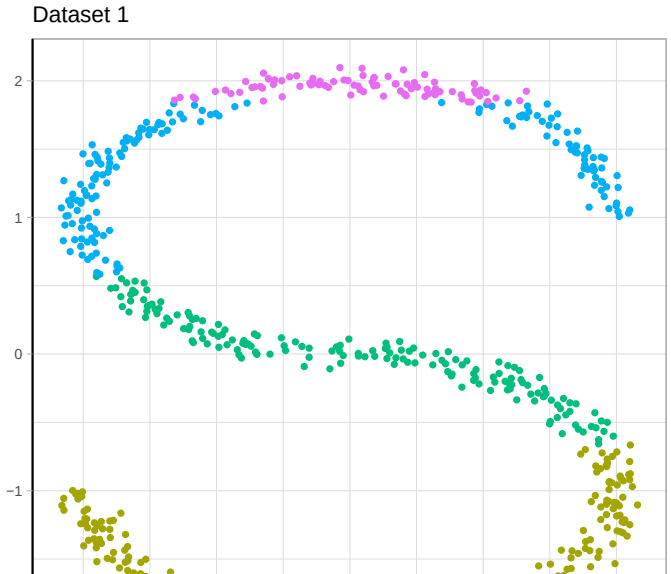
Linear Latent Tangent Space Alignment (LLTSA)

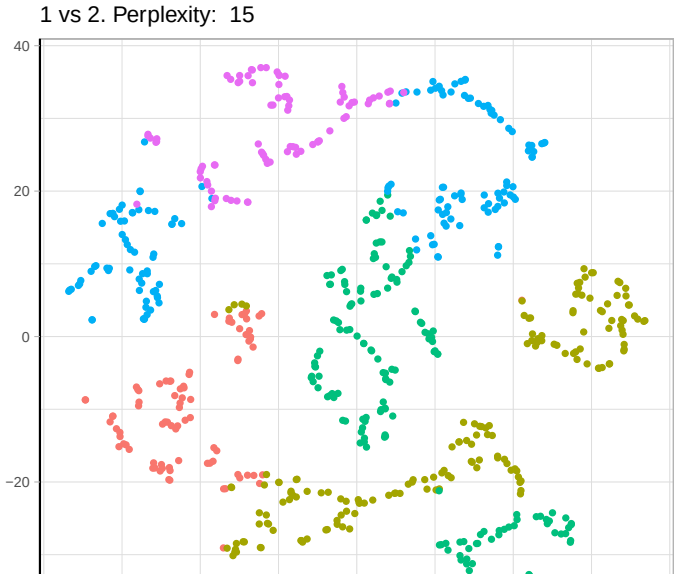
- LTSA varianthttps://www.sciencedirect.com/science/article/pii/S0925231206004577
- computes nearest neighbors, finds a tangent space to each neighborhood and combines those to find an embedding that aligns the tangents



t-SNE

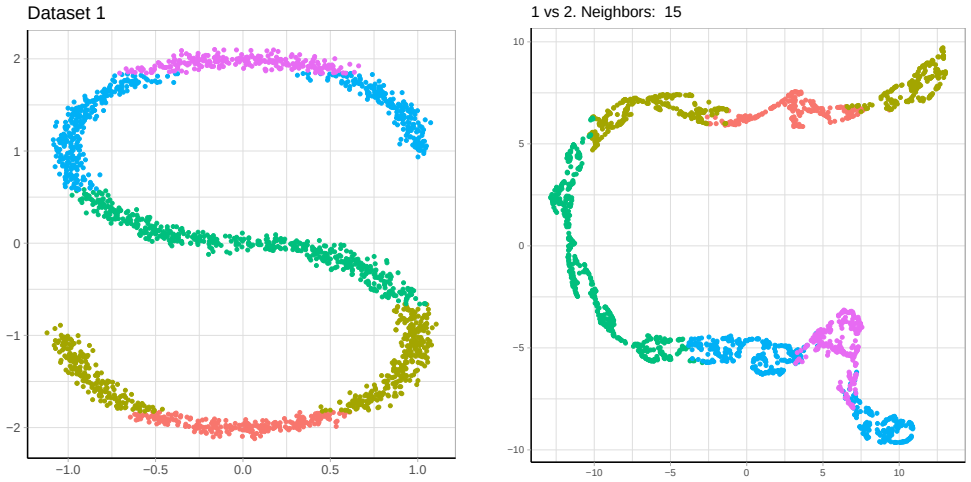
- Stochastic Neighbor Embedding varianthttp://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
- measures similarity between points by converting euclidian distances to probabilities according to the normal distribution, then tries to minimize the similarities in the projected space (using KLD) according to a t-distribution
- has a cost function that includes an attractive force and a repulsive one, optimized using gradient descent





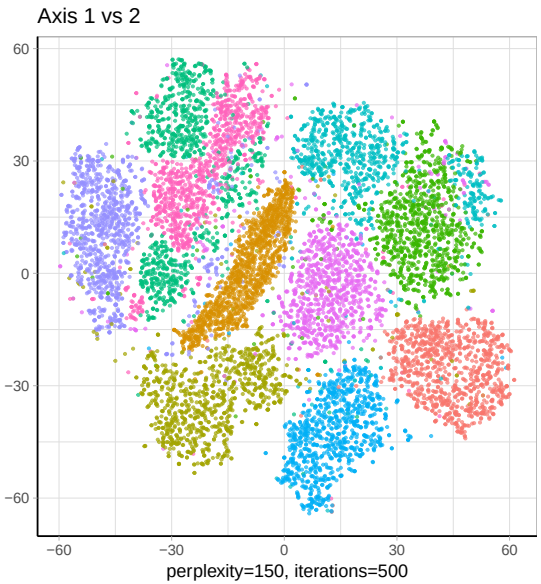
UMAP

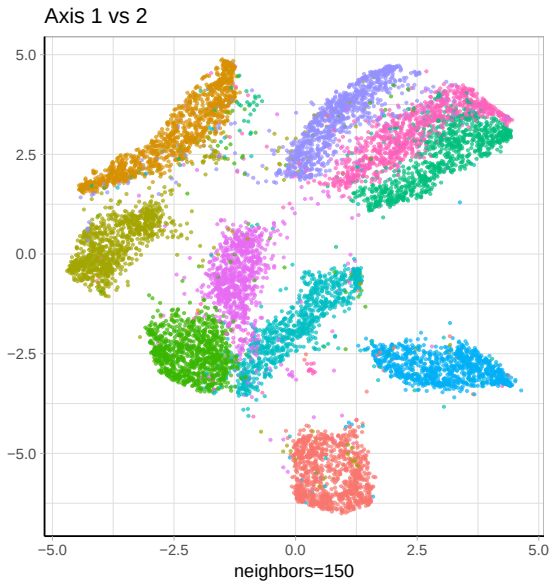
- Uniform Manifold Approximation and Projectionhttps://arxiv.org/abs/1802.03426
- much faster, has similar or better results than t-SNE and can transfer learning (can be used to predict)
- preserves more of the global structure



t-SNE vs UMAP

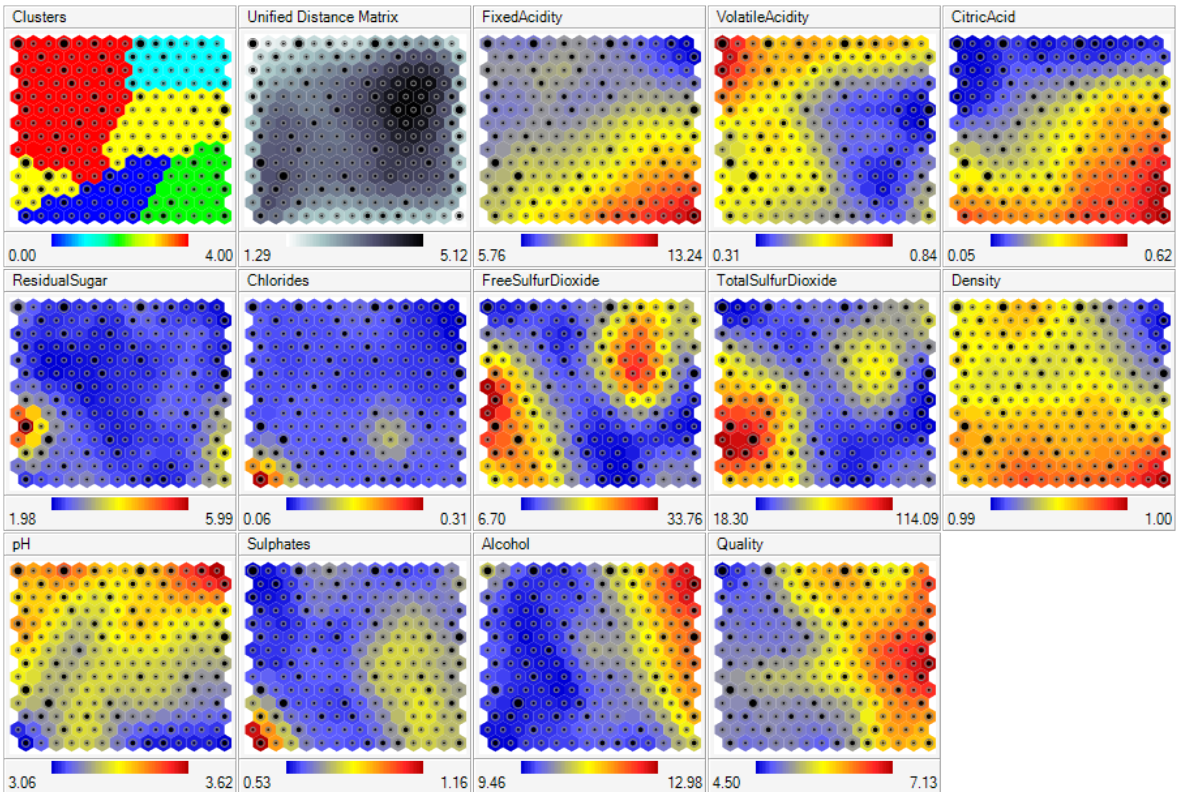
- MNIST dataset visualization





Self Organizing Maps (SOM)

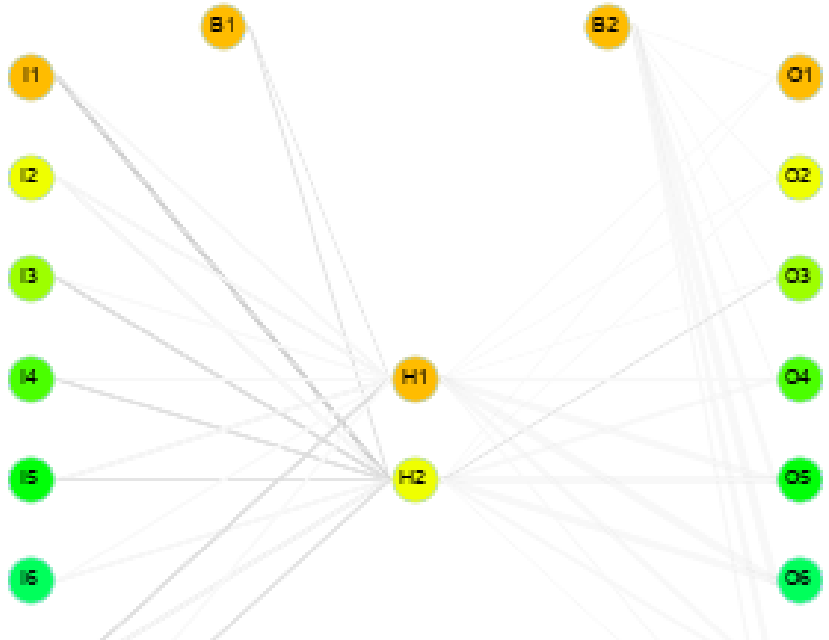
- a type of neural network mapping high dimensional data to a fixed grid of cells, preserving the topology
- very nice for data exploration, even when having few features (fallen out of favor recently, but remains a powerful technique)



Visualization done with Peltarion Synapse (unfortunately discontinued), still available as an online platform

Autoencoders

- easy to implement, harder to interpret
- all tunings of neural networks apply to autoencoders: topology, cost function, type of layers, type of cells, type of activation functions, number of nodes, regularization, skip connections, dropout/dropconnect, optimizers, network initialization
- use convolutions when there is some spatial relation, [maybe] recurrent networks for time-series



Autoencoders

- sample with highly correlated input features: hue coding with a simple one layer autoencoder
- 150 input features, 1000 samples

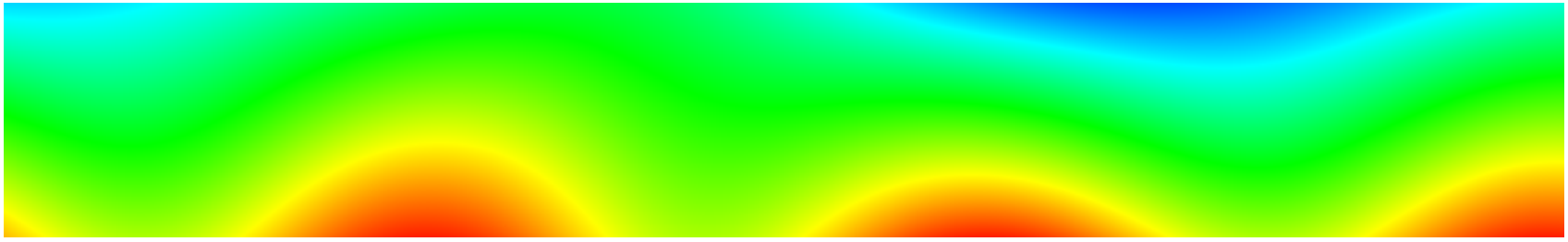
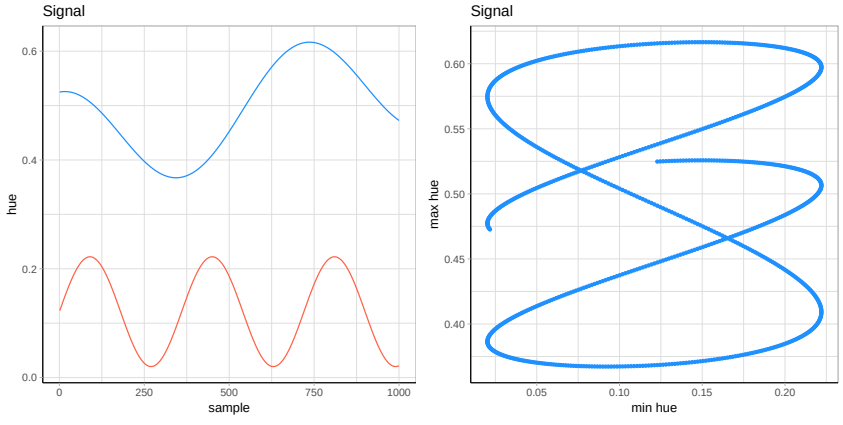
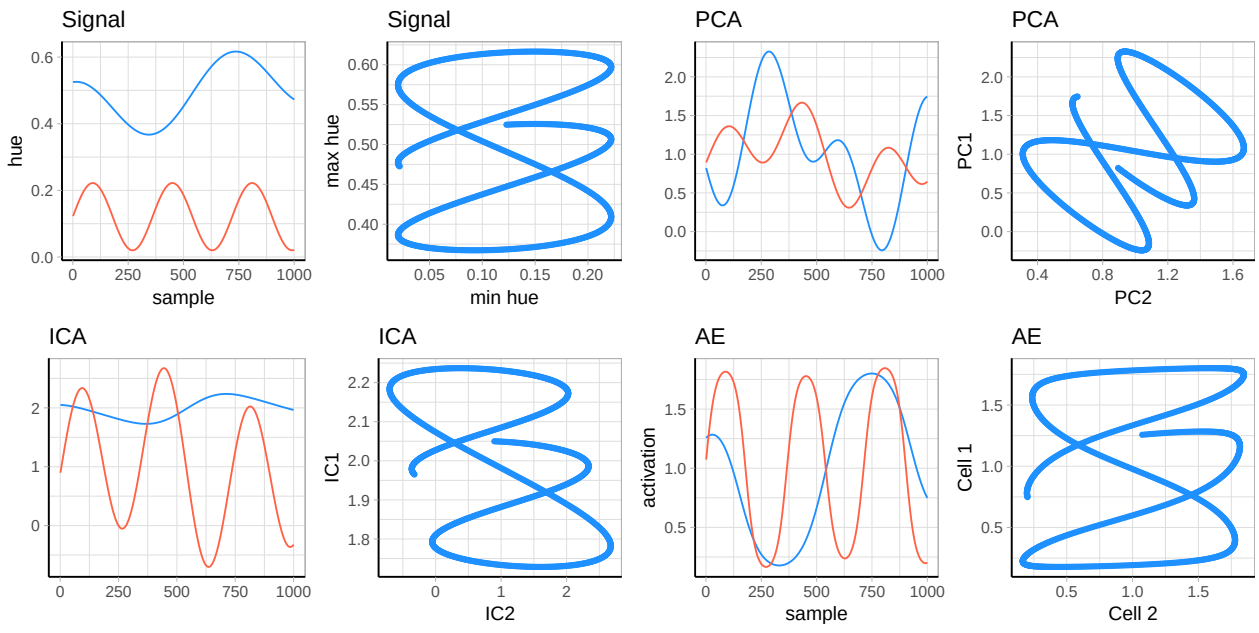


Figure 3:



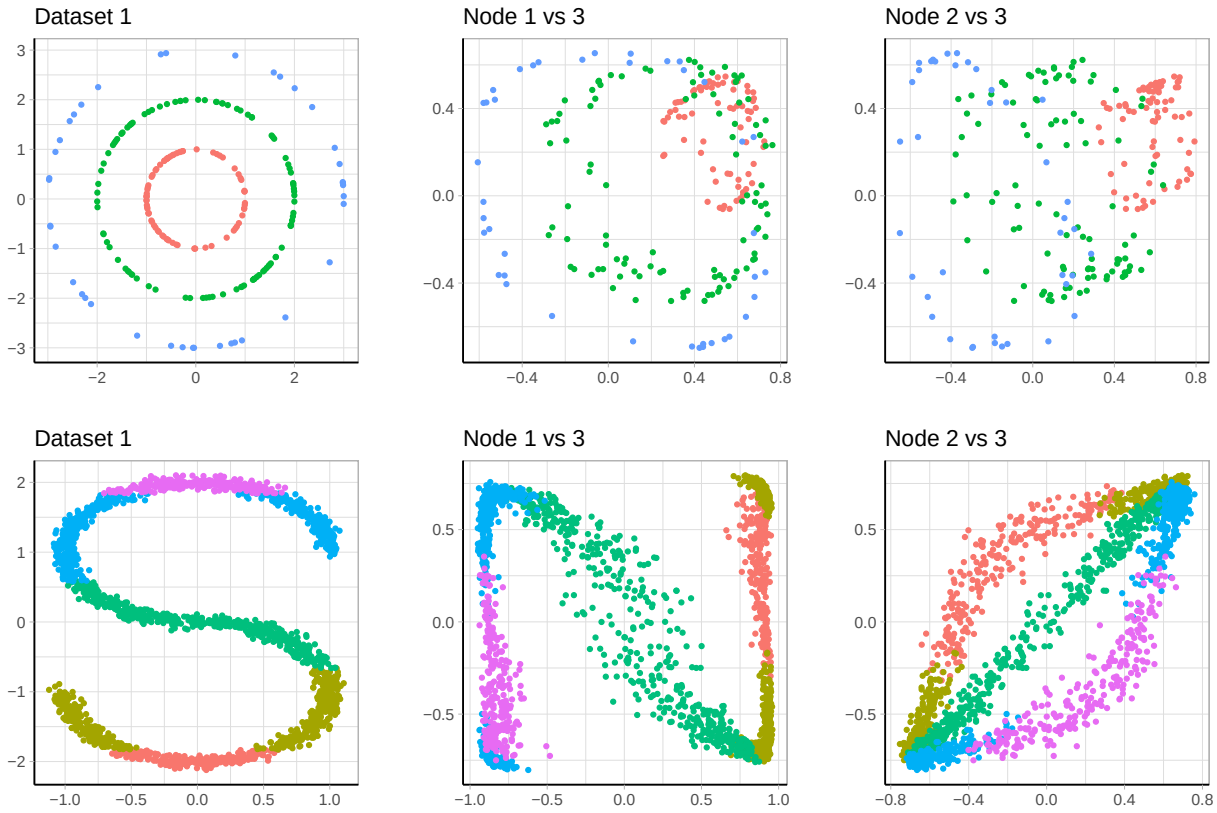
Autoencoders vs PCA vs ICA

- autoencoders are better at recovering the original features and the interaction between them
- why not always use autoencoders? sample size needed, overfitting dangers, lots of tuning needed, interpretability



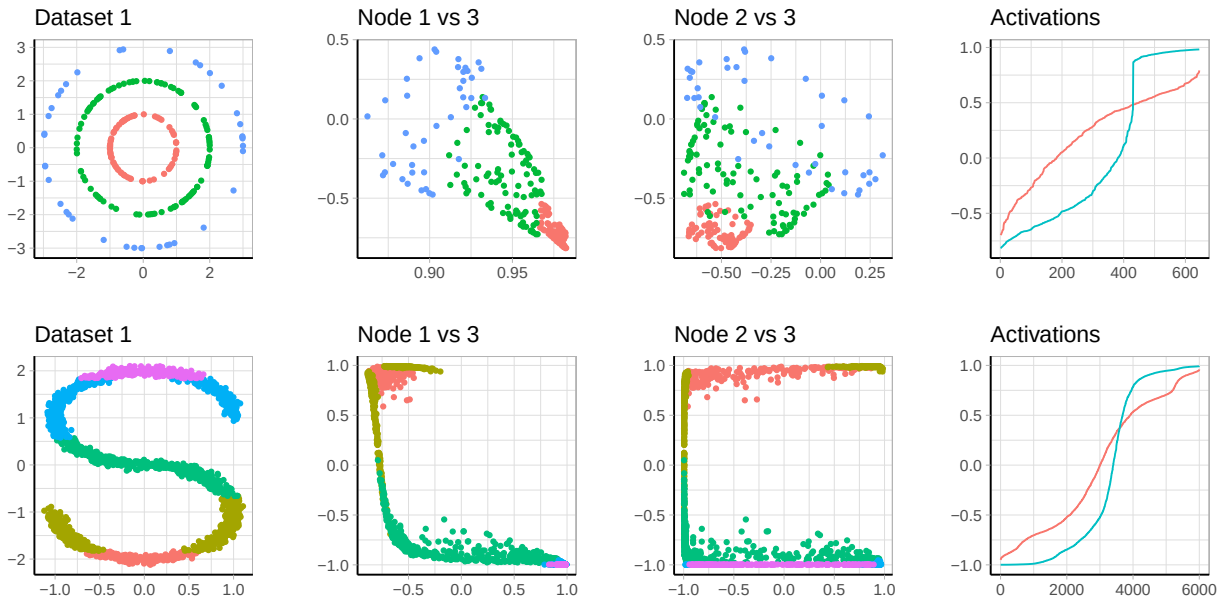
Deep Autoencoders

- multiple layers, usually with a symmetric hourglass topology
- latent layer in the middle



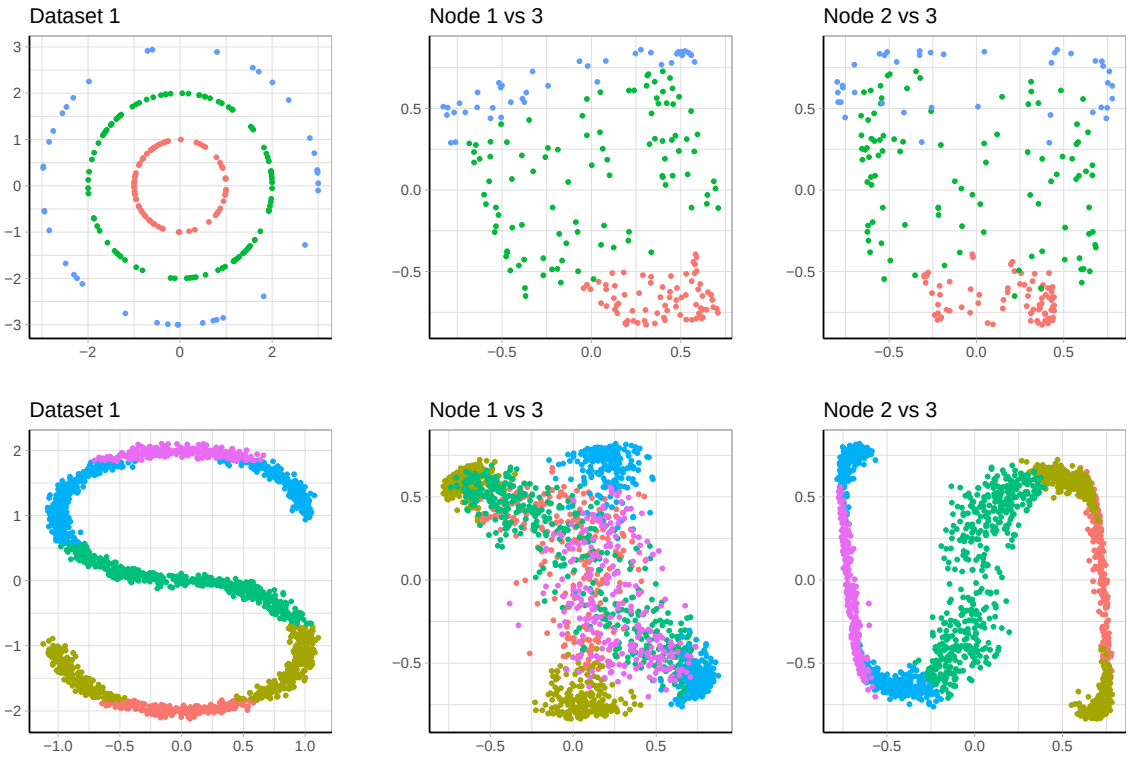
Sparse Autoencoders

- adds a penalty for multiple activations
- reconstruction results will be similar, the difference is in the sparsity of activations and, thus, robustness and interpretability
- theoretically forces better data separation by repelling different samples in the latent space



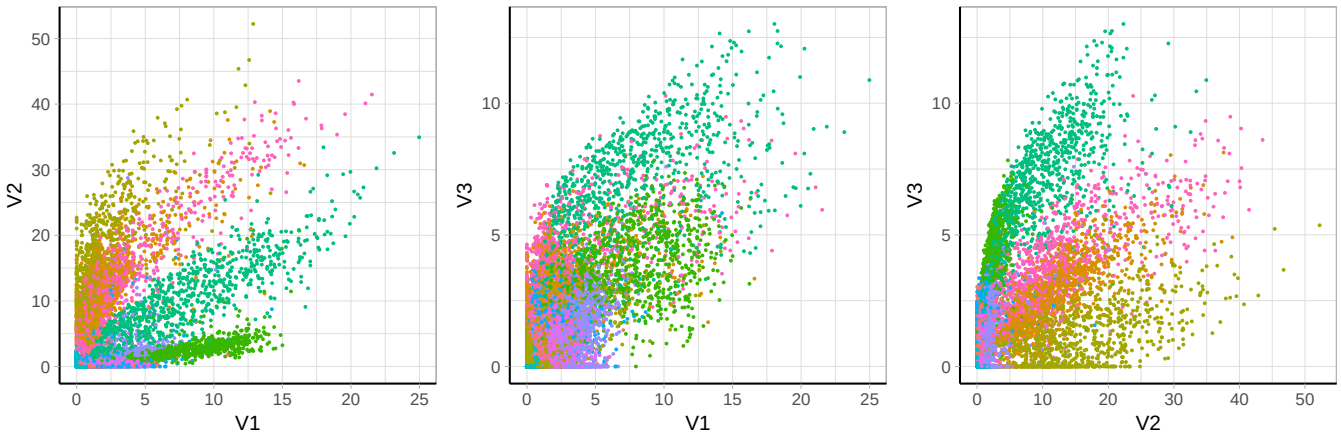
Stacked Autoencoders

- chaining encoders and decoders, can be seen as a series of resampling downward/upward



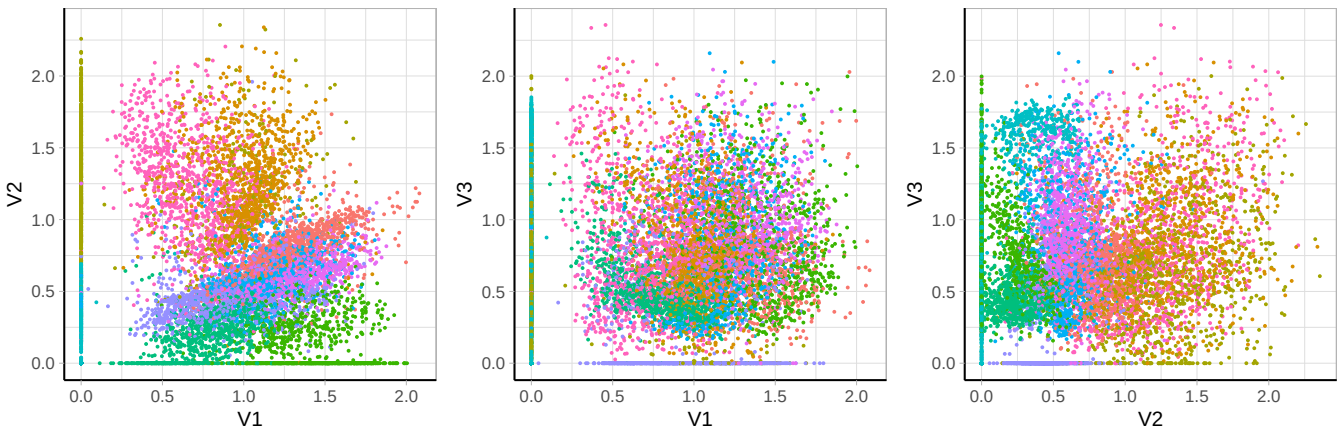
Denoising Autoencoders

- based on a simple data augmentation trick: add noise to inputs while keeping the originals as output
- increased robustness
- projections sample is for MNIST



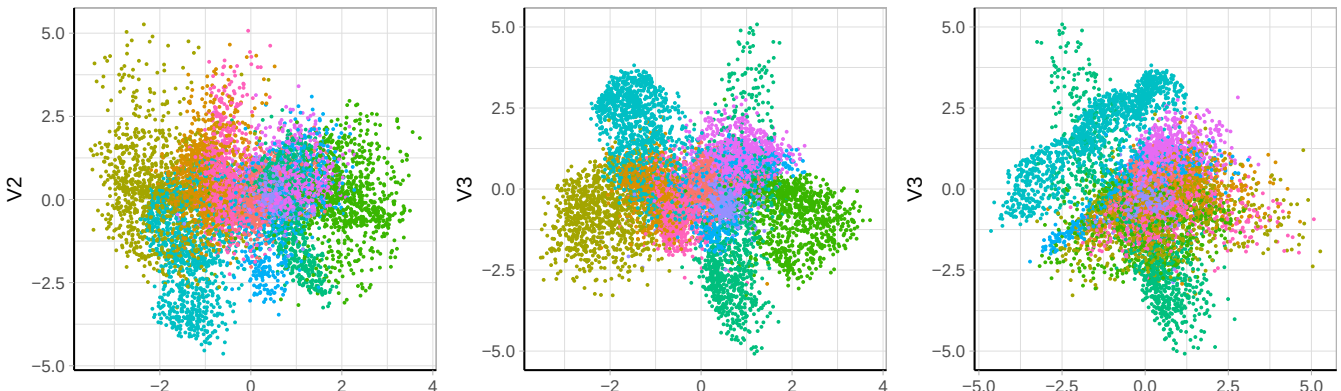
Contractive Autoencoders

- a type of regularized autoencoders
- only the loss function differs, it will penalize sensitivity of latent activations to inputs
- can be combined with other types (sample projection is for denoising-contractive AE)



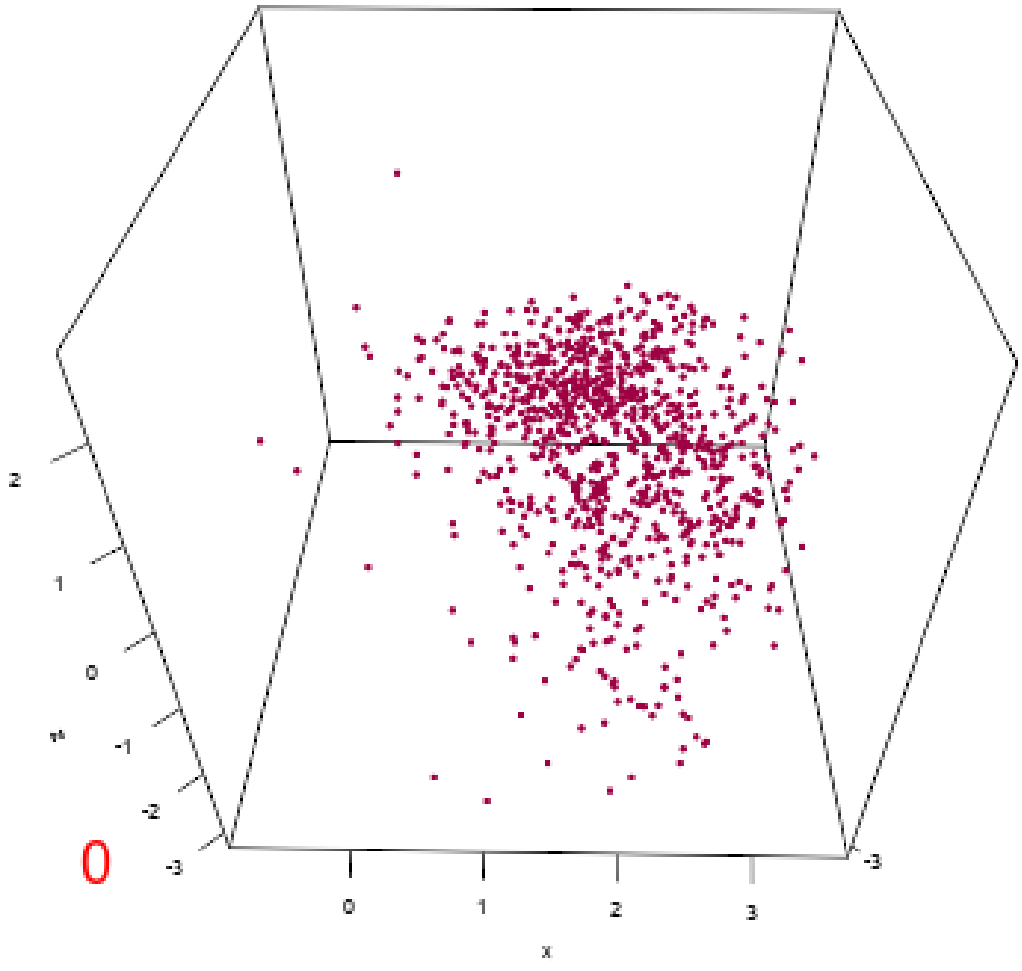
Variational Autoencoders (VAE)

- variational autoencoders - discovering latent factors via variational inference
- latent space is special, stores probability distributions (for each dimension a mean and a standard deviation is kept). Decoder samples from a normal distribution in the latent layer. The "reparametrization trick" allows the autoencoder to backpropagate
- semi-supervised extension: cVAEhttps://arxiv.org/abs/1406.5298



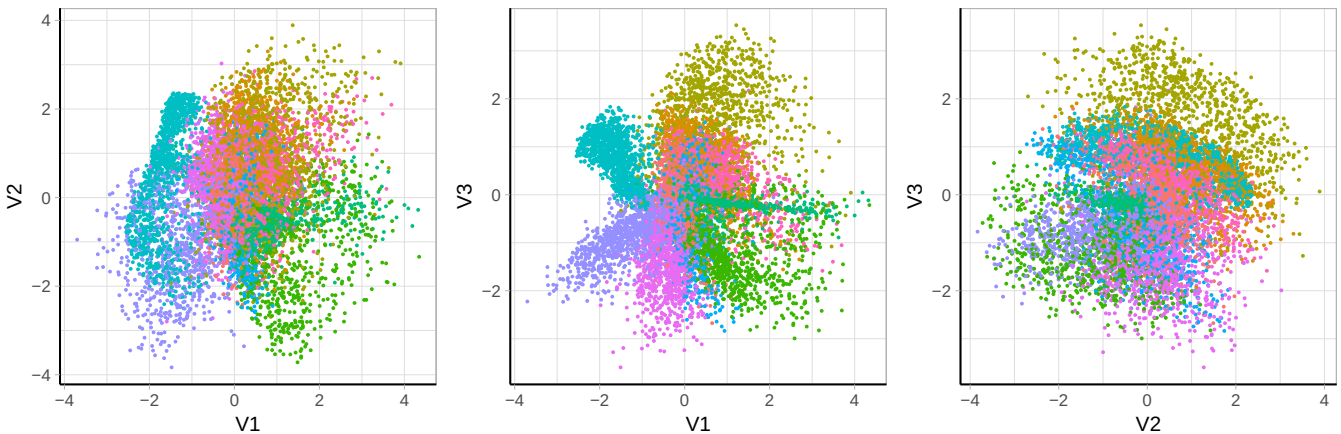
VAE

- latent space representation and manifold traversal (note: 2 layers and only 3 (x2) cells in the latent layer)



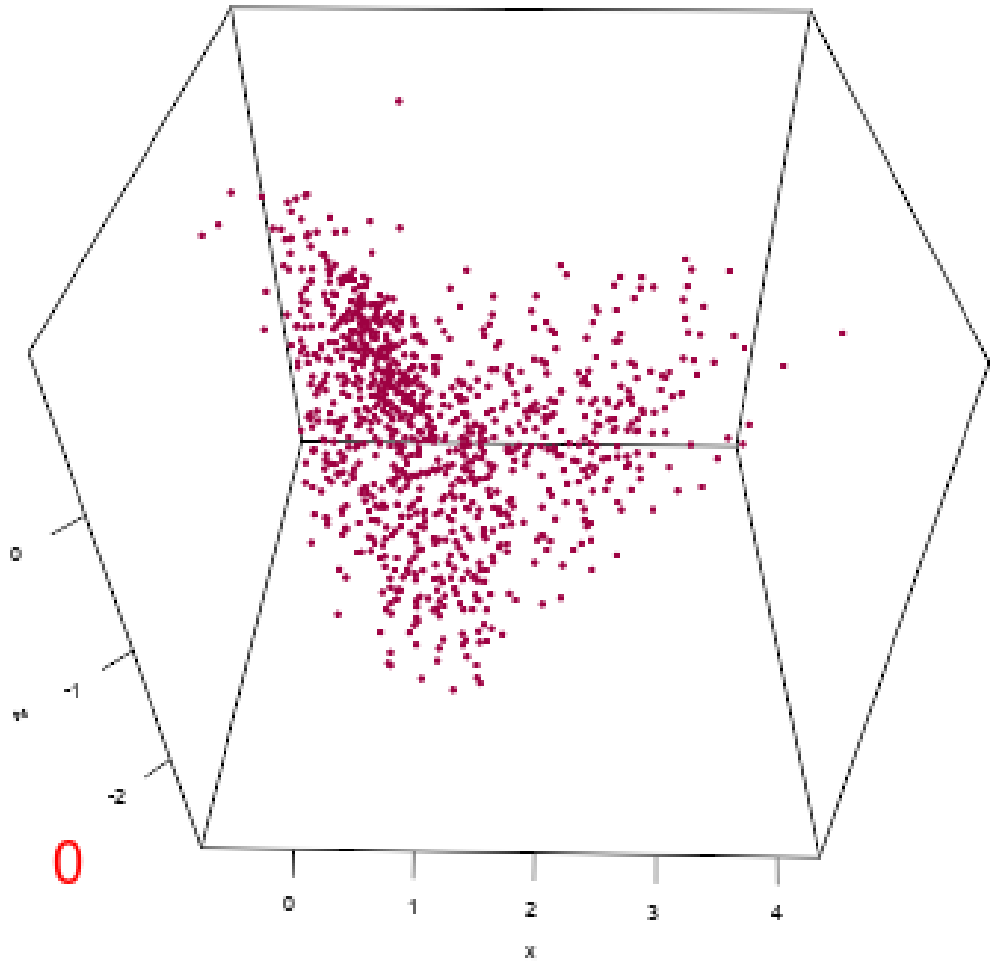
beta-VAE

- beta-VAEhttps://openreview.net/forum?id=Sy2fzU9gl: a step towards interpretability
- tries to disentangle the latent factors (in an ideal representation single latent units are sensitive to single generative factors)
- how beta works: simple multiplication in the VAE objective function, which seems to encourage the latent representation axes to be sensitive to changes in generative factorshttps://arxiv.org/abs/1804.03599



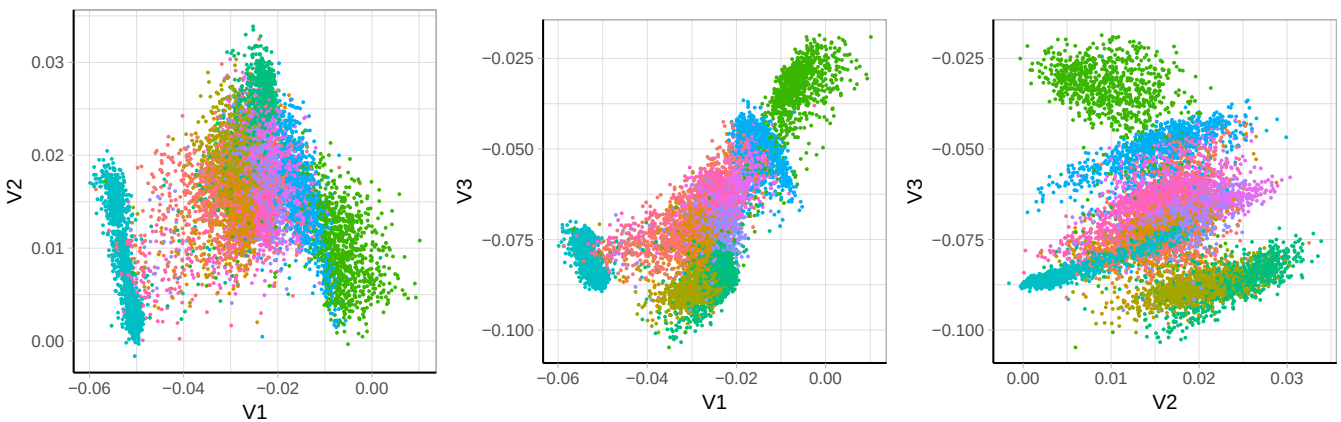
beta-VAE

- activations and traversal for the MNIST dataset: 2 layers network and only 3 (x2) cells in the latent layer



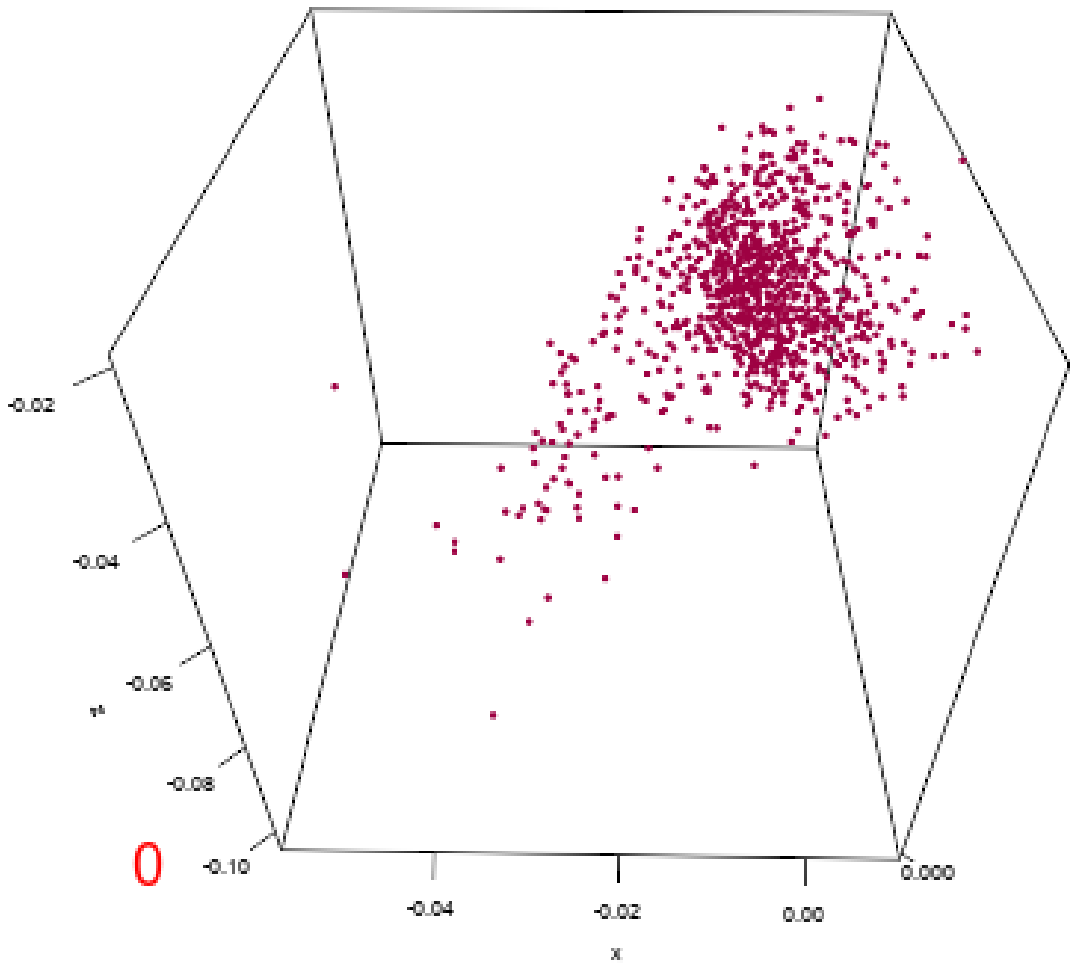
cVAEGAN

- Generative Adversarial Networks: great for generating plausible highly detailed data from the distribution, while autoencoders learn averaged representations
- cVAEGAN: semi-supervised, conditional learning
- combining multiple networks, some with common layers: encoder, decoder, classifier and discriminator



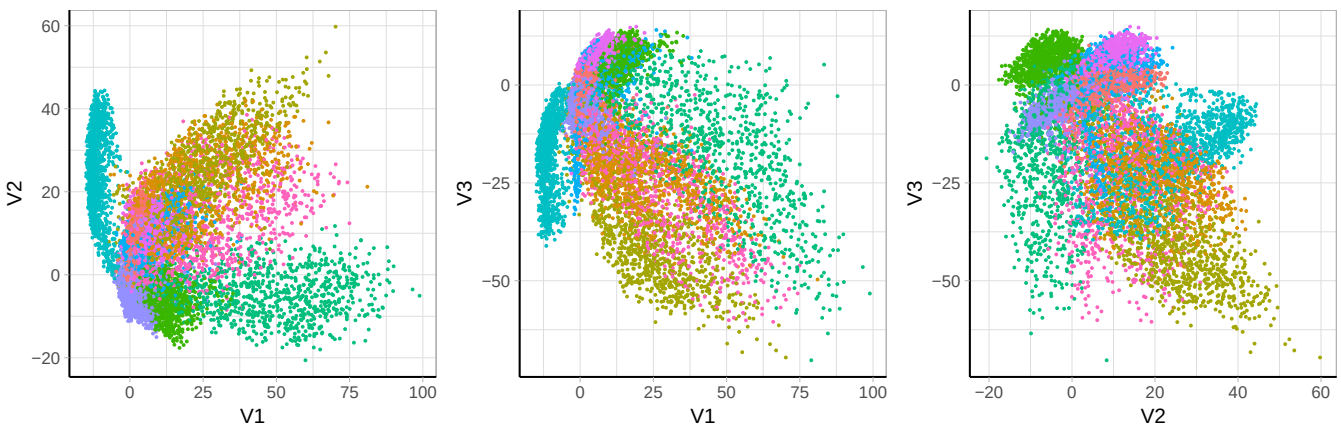
cVAEGAN

- convolutional layers, only 3 cells in the latent layer



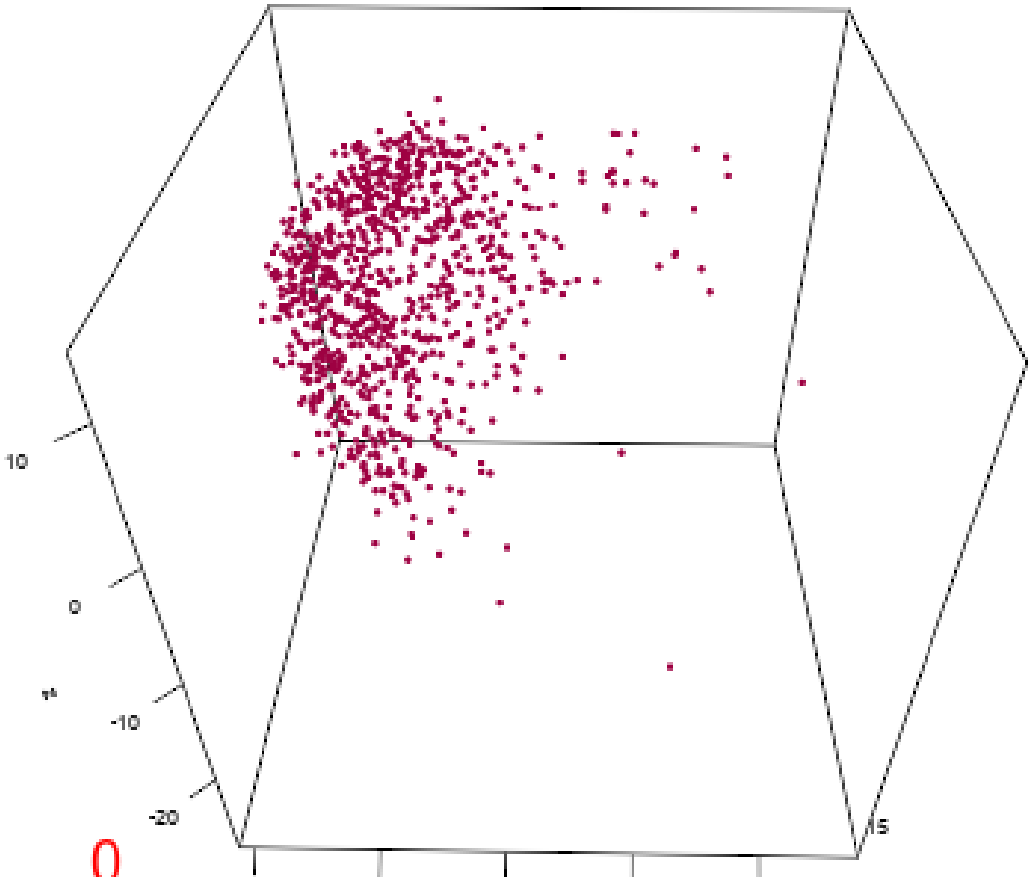
Uncertainty Autoencoder (UAE)

- Uncertainty Autoencodershttps://arxiv.org/abs/1812.10539: adding noise in the latent layer
- tries to find a sparse representation of the original data that best allow reconstruction, maximizing the mutual information between the data points and a noisy representation in the latent layer



UAE

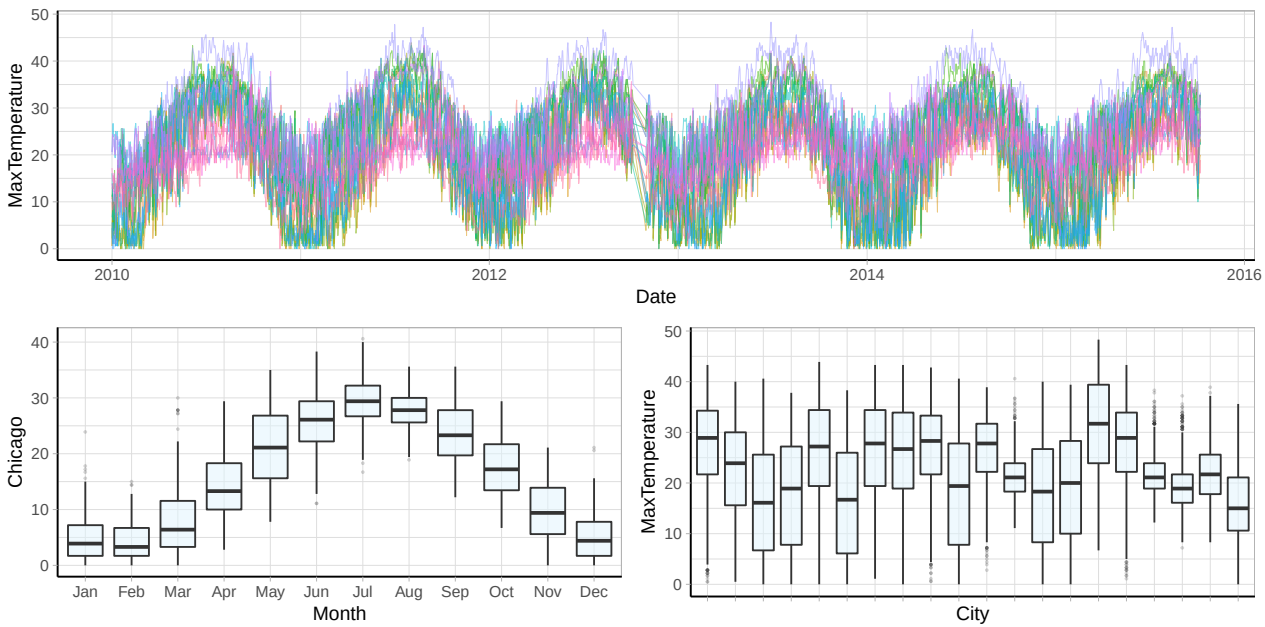
- data separation is already good, although reconstructions don't look so well
- sample network is small, embedding more knowledge in it can help: deeper networks, more neurons, convolutions



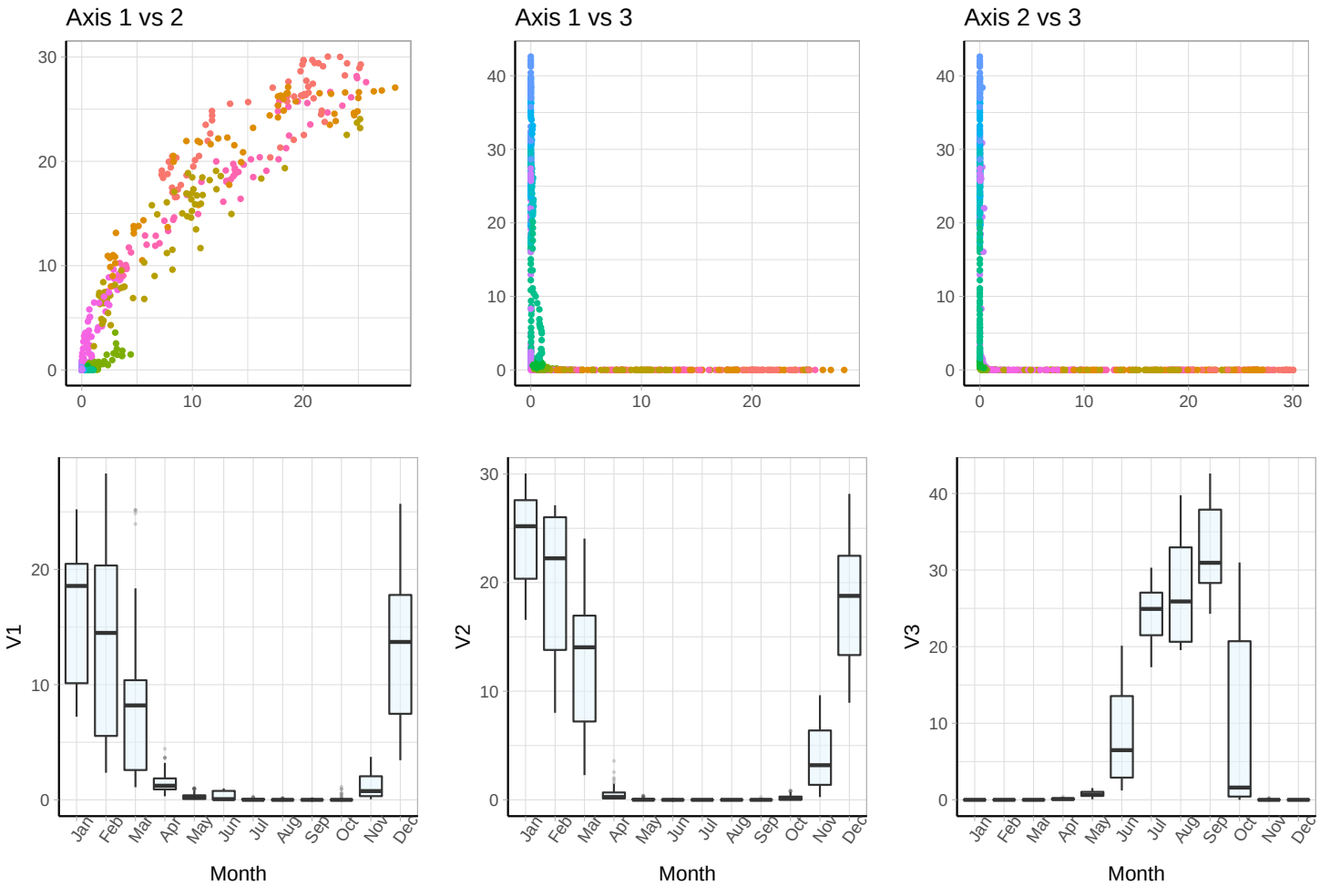
Recurrent autoencoders

- the input and the output is a sequence
- using special cells: GRU, LSTM, Nested LSTM
- long term dependencies problem: Bi-LSTM, attention. Maybe "attention is all you need".

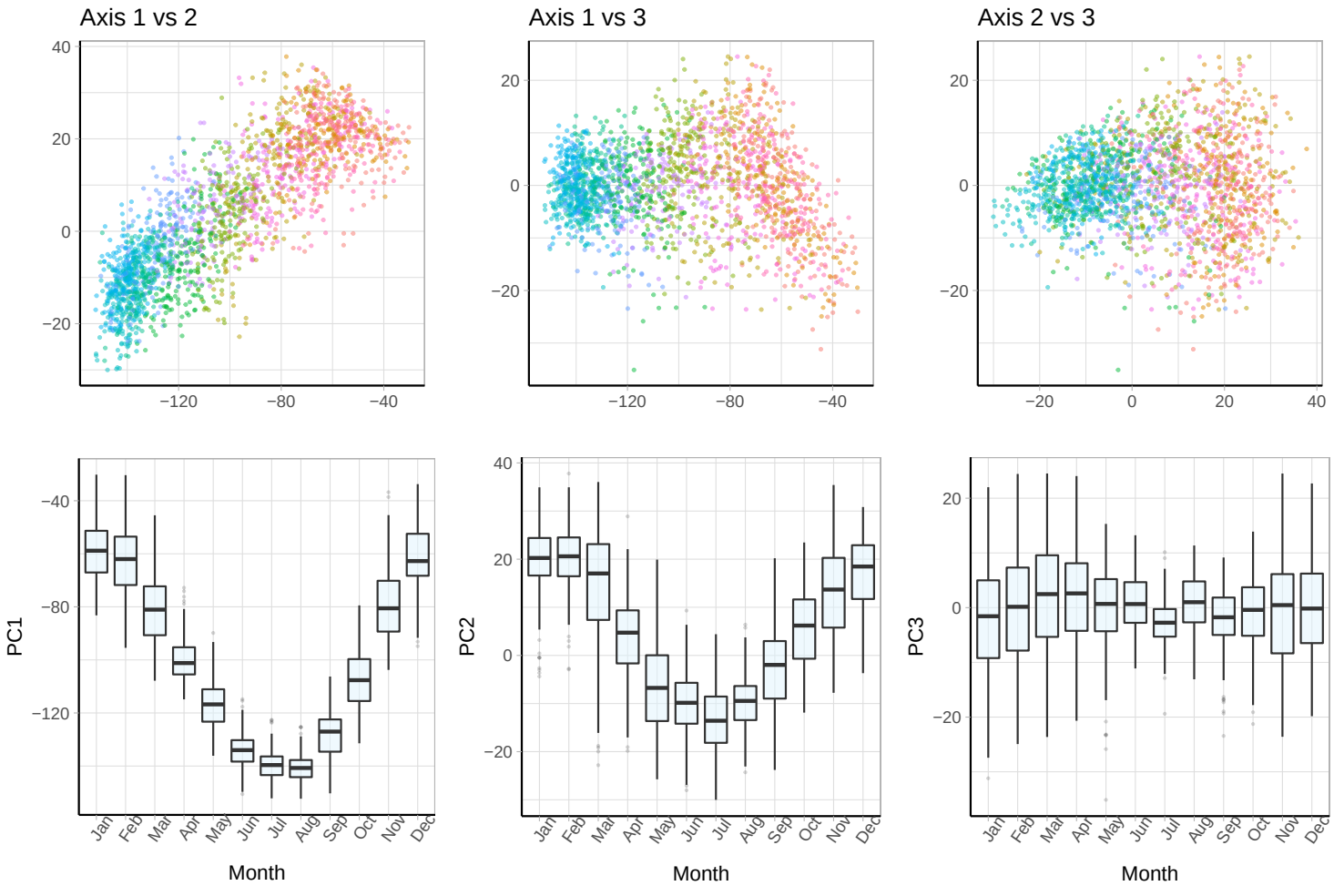
Data source: https://www.ncdc.noaa.gov/ghcn-daily-description



Recurrent autoencoders



PCA again


Axis 1 vs 2
Axis 1 vs 3
Axis 2 vs 3
PC1
PC2
PC3
Month
Month
Month
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

Many more

- Nonnegative Matrix Factorization (NMF)
- Neighbourhood Components Analysis (NCA)http://www.cs.toronto.edu/~fritz/absps/nca.pdf
- Maximum Variance Unfolding (MVU)http://cseweb.ucsd.edu/~saul/papers/nldr_aaai06.pdf
- Generative Topographic Mappinghttps://www.microsoft.com/en-us/research/wp-content/uploads/1998/01/bishop-gtm-ncomp-98.pdf
- Diffusion Mapshttp://inside.mines.edu/fs_home/whereman/talks/delaPorte-Herbst-Hereman-vanderWalt-DiffusionMaps-PRASA2008.pdf
- Twin Kernel Embedding (TKE)https://www.researchgate.net/publication/5289090

Many more

- Deep Belief Networks and Deep Bolzmann Machines as autoencoders alternative
- Conditional Subspace VAE (CSVAE)https://arxiv.org/abs/1812.06190
- Vector Quantised VAE (VQ-VAE)https://arxiv.org/abs/1711.00937
- Total Correlation VAE (beta-TCVAE)https://arxiv.org/abs/1802.04942
- Independent Subspace Analysis VAE (ISA-VAE)https://openreview.net/forum?id=rJl_NhR9K7
- Factorized Action VAE (FAVAE)https://arxiv.org/abs/1902.08341
- FactorVAEhttps://arxiv.org/abs/1802.05983
- oi-VAEhttps://arxiv.org/abs/1802.06765
- Auto-Classifier-Encoder (ACE)https://arxiv.org/abs/1508.06585
- InfoGANhttps://arxiv.org/abs/1606.03657
- Adversarial Information Factorization