Deep Learning

12. Unsupervised learning: clustering, autoencoders, VAEs, GANs

Viacheslav Dudar

Taras Shevchenko National University of Kyiv

2018

Why unsupervised learning?

- No labels during training
- Aim: describe hidden structure of unlabeled data
- Humans are learning in unsupervised way
- Has close relation with reinforcement learning

Types of unsupervised learning

- Probability density estimation
- Clustering
- Representation learning
- Dimensionality reduction
- Generating new samples

Clustering methods

Main types of clusterization:

- Centroid models: each cluster is represented by a single mean vector (k-means)
- Connectivity models: models based on distance connectivity (hierarchical clustering)
- Distribution models: clusters are modeled using statistical distributions
- Density models: clusters as connected dense regions in the data space (DBSCAN, OPTICS)
- Many more: $>$ 100 clusterization algorithms

k-means

Dataset: $\{x_1, \ldots x_n\}$
Cluster centers: $\{\mu_k, k = 1 \ldots K\}$
Binary indicator variables r_{nk}: if x_n is assigned to cluster k then $r_{nk} = 1$ and $r_{nj} = 0$ for $j \neq k$.
We want to minimize:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| x_n - \mu_k \right\|^2 \rightarrow min$$

k-means

Minimize J via two-stage procedure:
1) Assign each point to closest cluster center.

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

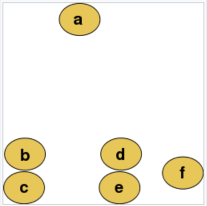
2) Set μ_k as centers of clusters:

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}.$$

DBSCAN

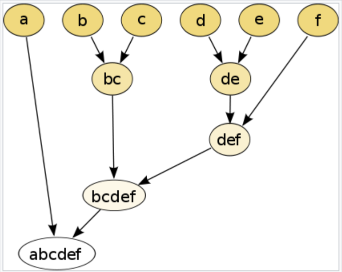
1) Find the points in the ϵ neighborhood of every point, and identify the core points with more than minPts neighbors.
2) Find the connected components of core points on the neighbor graph, ignoring all non-core points.
3) Assign each non-core point to a nearby cluster if the cluster is an ϵ neighbor, otherwise assign it to noise.

Hierarchical clustering



Raw data

The hierarchical clustering dendrogram would be as such:



Principal components analysis

First principal component:

$$\text{maximize} \frac{1}{2N} \sum_{n=1}^{N} (u_1^T x_n - u_1^T \bar{x}_n)^2$$

$$= u_1^T S u_1$$

i.e., variance of the projected data

where the sample mean and covariance are given by:

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

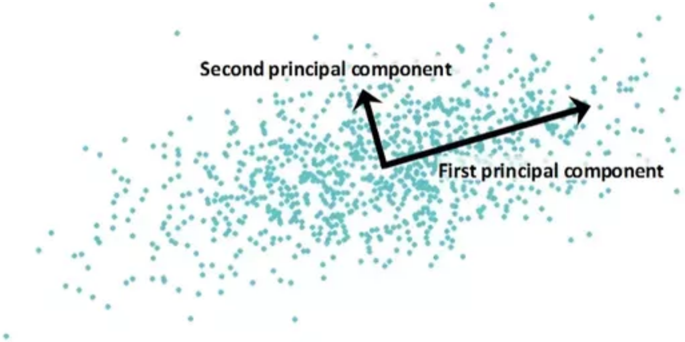
$$S = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T$$

PCA

Second principal component:

$$\begin{aligned} \text{maximize} \quad & u_2^T S u_2 \\ \text{subject to} \quad & ||u_2|| = 1 \\ & u_2^T u_1 = 0 \end{aligned}$$

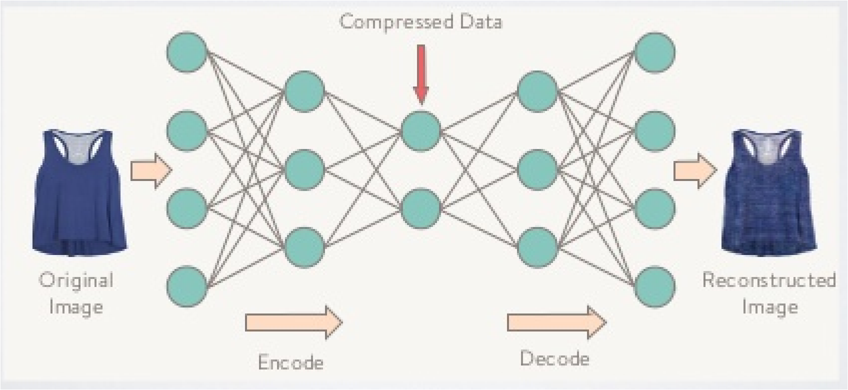
k-th principal component is equal to eigenvector of S that corresponds to k-th largest eigenvalue of S.



Classical approaches

- Graphical models: Bayesian networks (directed graphs), Markov random fields (undirected graphs)
- Mixture Models and EM (K-means clustering, Mixtures of Gaussians)
- Approximate inference (variational inference, expectation propagation)
- Sampling methods (rejection sampling, Markov chain Monte-Carlo, Gibbs sampling)
- Continuous latent variables (principal component analysis, probabilistic PCA, kernel PCA, nonlinear latent variable models)

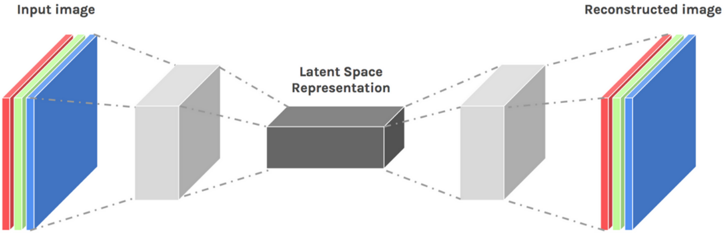
Autoencoders



Autoencoders properties

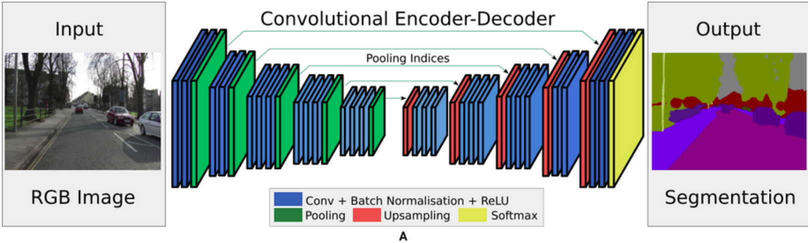
- Target output is input
- Linear decoder $\rightarrow$ equivalent to PCA
- Learning low-dimensional data representation

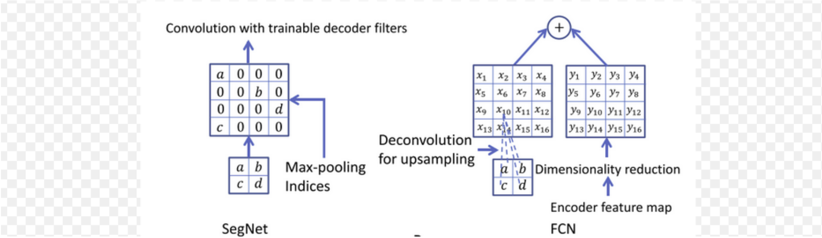
Convolutional autoencoder



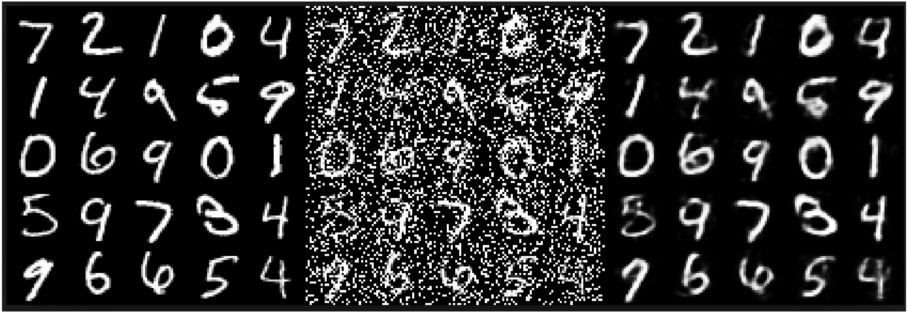
Autoencoder-like supervised learning tasks

Segnet: encoder-decoder net for segmentation

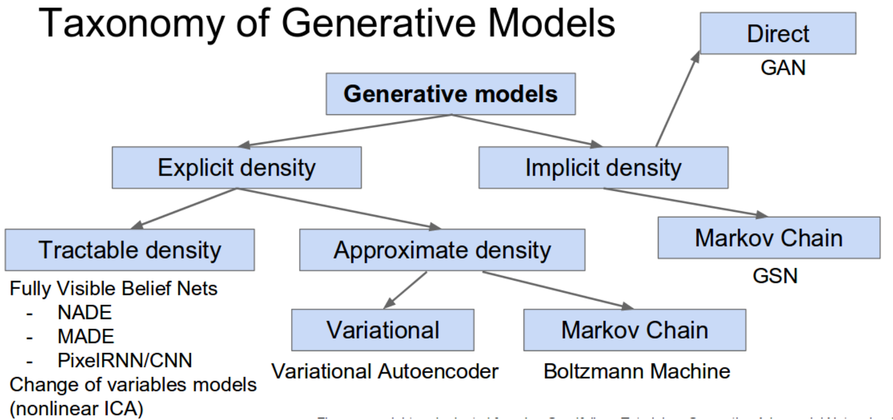




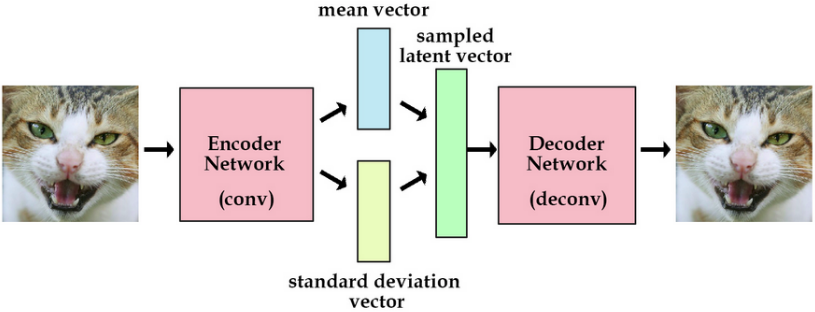
Denoising autoencoder



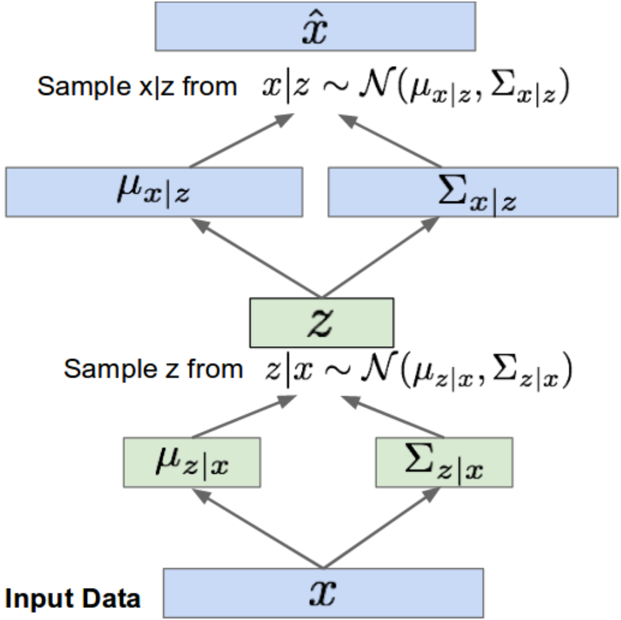
Generative models



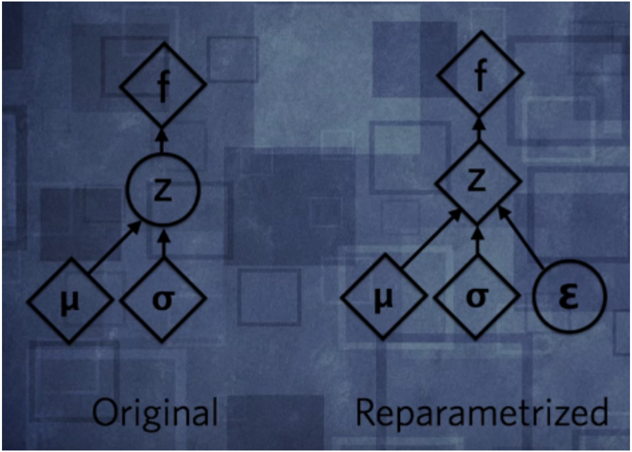
Variational autoencoder



Variational autoencoder: more details



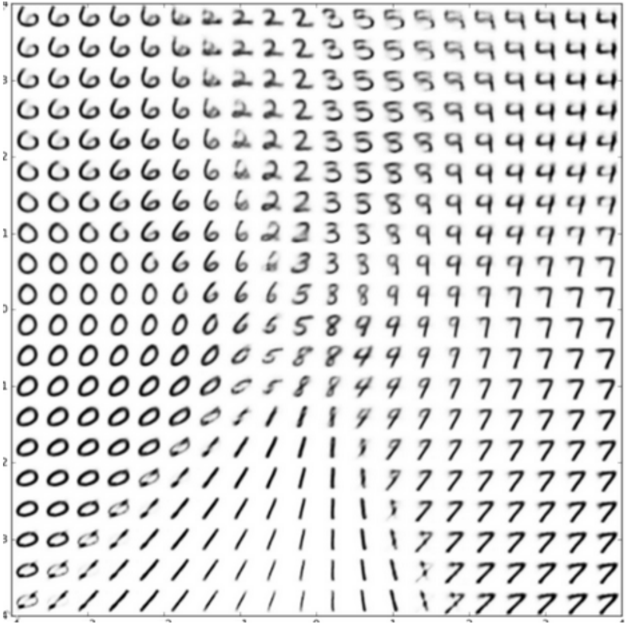
VAE: reparametrization



VAE: properties

- Minimizing reconstruction loss plus KL divergence between distribution of latent vector z and unit normal distribution
- To generate new examples we sample z from unit normal distribution and run through encoder network
- Has interpretation in terms of variational maximization of lower bound of log-likelihood

VAE: generating digits



VAE: generating faces

PixelRNN

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Likelihood of image x

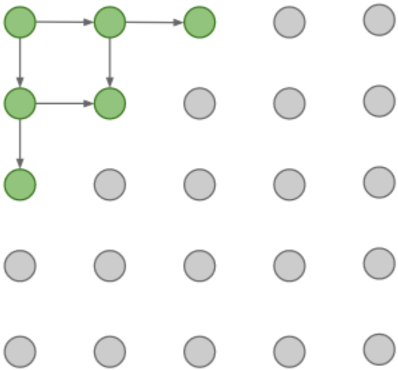
Probability of i'th pixel value given all previous pixels

Will need to define ordering of "previous pixels"

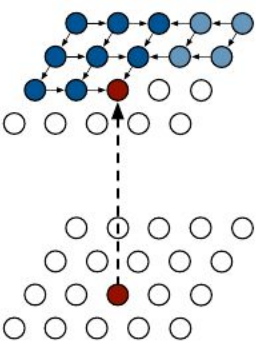
Complex distribution over pixel values => Express using a neural network!

Then maximize likelihood of training data

PixelRNN



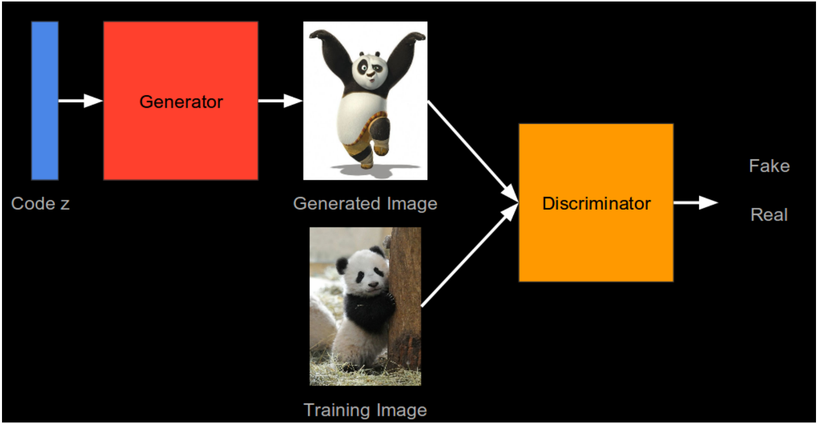
Diagonal BLSTM:



PixelRNN

Figure 1. Image completions sampled from a PixelRNN.

Generative adversarial network



Generative adversarial network

Generative Adversarial Network (GAN)

Objective function:

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[1 - \log D(G(\mathbf{z}))]$$

For each iteration:

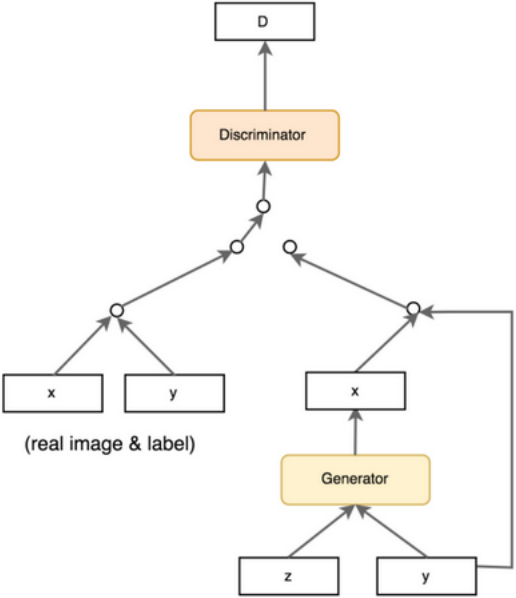
- Sample a mini-batch of fake images and true images
- Update G using back-prop
- Update D using back-prop

Very difficult to optimize:

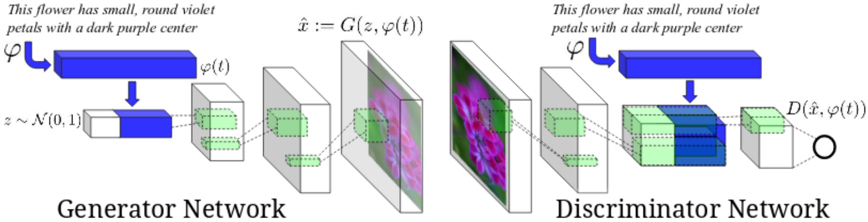
- Min-max problem: finding a saddle point instead of a local optimum, unstable

GAN: face generation

Conditional GAN



Text to Image synthesis



Text to Image synthesis

Caption	Image
a pitcher is about to throw the ball to the batter	
a group of people on skis stand in the snow	
a man in a wet suit riding a surfboard on a wave	

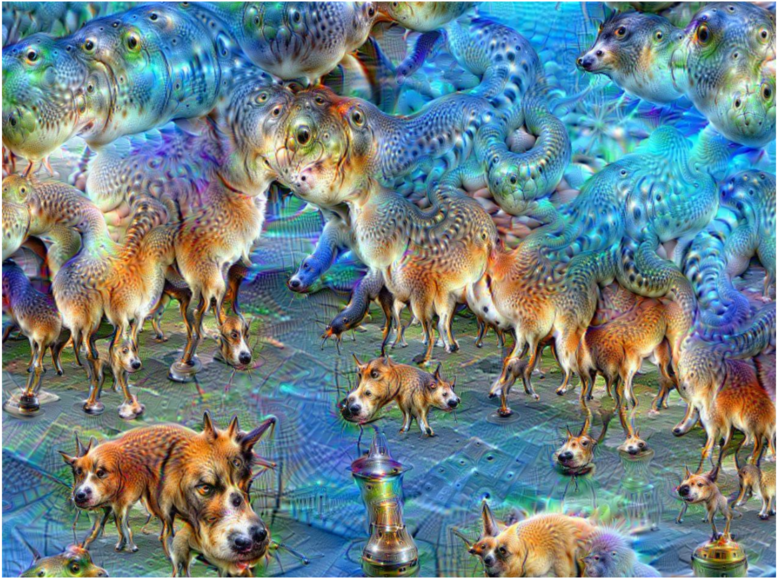
Text to Image synthesis

This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face

This bird is white with some black on its head and wings, and has a long orange beak

This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments

Deep dream



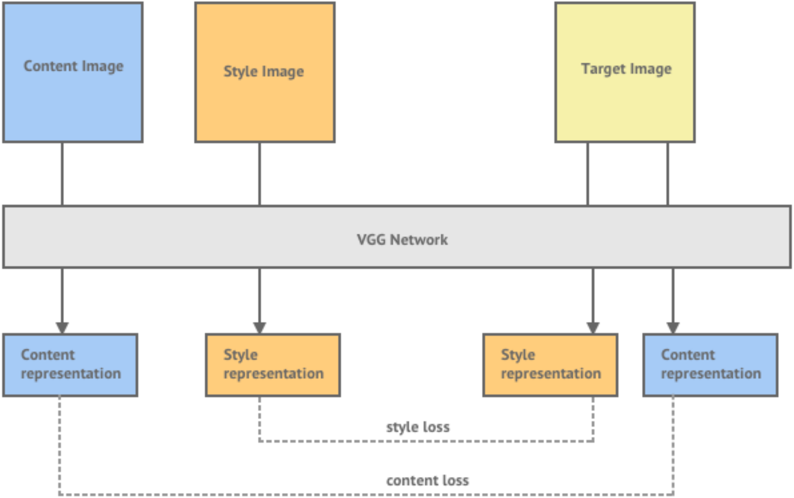
Style transfer

$$G_{ij} = \sum_k F_{ik} F_{jk}$$

$$\mathcal{L}_{style} = \frac{1}{2} \sum_{l=0}^{L} (G_{ij}^l - A_{ij}^l)^2$$

$$\mathcal{L}_{content} = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

Style transfer



Style transfer example

Read more

Deep generative models:
http://www.cs.toronto.edu/~urtasun/courses/CSC2541_Winter17/Deep_generative_models.pdf
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf
Style transfer:
https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf
Google dream:
https://www.youtube.com/watch?v=dbQh1I_uvjo
Some classical approaches to modeling probability distributions (no neural nets): Bishop, Pattern Recognition and Machine Learning, Chapters 8 - 12