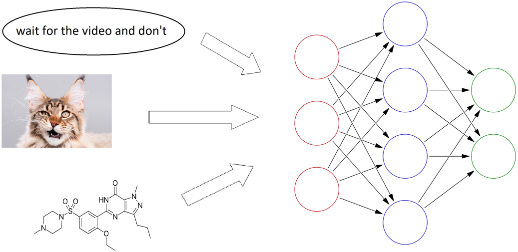
Deep processing of structured (non-vector) data

Łukasz Maziarka, Aleksandra Nowak, Łukasz Struski, Marek Śmieja

GMUM Group of Machine Learning Methods,
Jagiellonian University, Kraków

Structured / non-vector data

- Typical machine learning models assume that data are represented as vectors of fixed size.
- In practice, raw data often do not have a vector form:
  - texts
  - images of varied sizes
  - graphs
  - sets with varied sizes
  - data with missing attributes



Problem

"Structured data" – data which are not given as vectors of fixed dimension.

Question

How to process structured data by neural networks?

In details:

- how to represent structured data?
- how to process these representations?

Outline of workshop

PART I (missing data):

- representing missing data by probability measures
- processing of measures

M. Śmieja et al., *Processing of missing data by neural networks* NIPS 2018

Part II (other structured data):

- processing of sets
- application in processing of text, graphs, images

Ł. Maziarka, et al., *Deep processing of structured data*, arXiv:1810.01868, 2018,

M. Zaheer, et al. *Deep sets*, NIPS 2017

Part I

Processing of missing data by neural networks.

Motivation

Learning from incomplete data is one of the fundamental challenges in machine learning (Goodfellow et al., 2016):

- In **medical diagnosis**, some medial tests are expensive or invasive and, in consequence, only selected measurements are usually available for a given patient.
- In **industry**, the absence of sensor's measurement may be caused either by random factors, but can also indicate oncoming failure of the system.
- In **image processing**, part of the picture may be hidden or destroyed.

Typical approaches

Typical approaches:

- **deletion**: removing data with missing attributes – reduces information
- **completion (imputation)**: estimating missing values from observed ones – inserts unreliable information

Question

How to learn neural networks from incomplete data directly?

Contribution

Our framework[1]:

- can be combined with various types of networks
- requires a minimal modification in the architecture
- does not need complete data for training
- is theoretically justified

[1] M. Śmieja et al., *Processing of missing data by neural networks*, NIPS 2018

Missing data

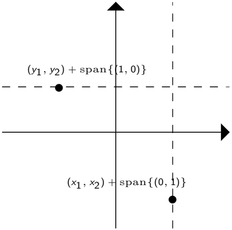
Missing data point is the affine subspace:

$$S = \mathrm{Aff}[x, J] = x + \mathrm{span}(e_J),$$

where:

- $x \in \mathbb{R}^D$ is a representative
- $J \subset \{1, \ldots, D\}$ is a set of missing attributes

Input:
$(x_1, ?)$
$(?, y_2)$

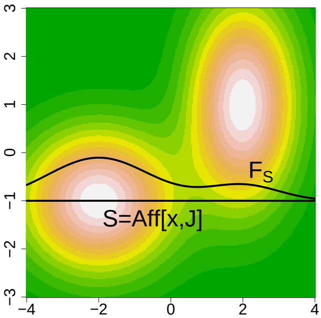


This is not very informative.

Density model

- F – probability distribution, which generates values at missing attributes
- We can model unobserved values of $S = \mathrm{Aff}[x, J]$ by restricting F to the affine subspace S:

$$F_S(x) = \begin{cases} \frac{1}{\int_S F(s)ds} F(x), \text{ for } x \in S, \\ 0, \text{ otherwise}, \end{cases} \tag{1}$$



Processing of densities

Assume we have a set of densities $\{F_S\}_S$ representing missing data.

Question

How to process density F_S by a neural network?

Possible answer

Idea:

- draw samples from F_S
- process this batch by the network
- take the expectation at the end of the network.

More formally

The network cost

$$\mathbb{R}^D \ni x \to \mathrm{cost}_\omega(x) \in \mathbb{R}$$

is replaced by

$$E[\mathrm{cost}_\omega(x) | x \sim F_S].$$

Our answer

Idea:

- take the expectation of neuron's response at first layer
- leave the rest of architecture unchanged

More formally

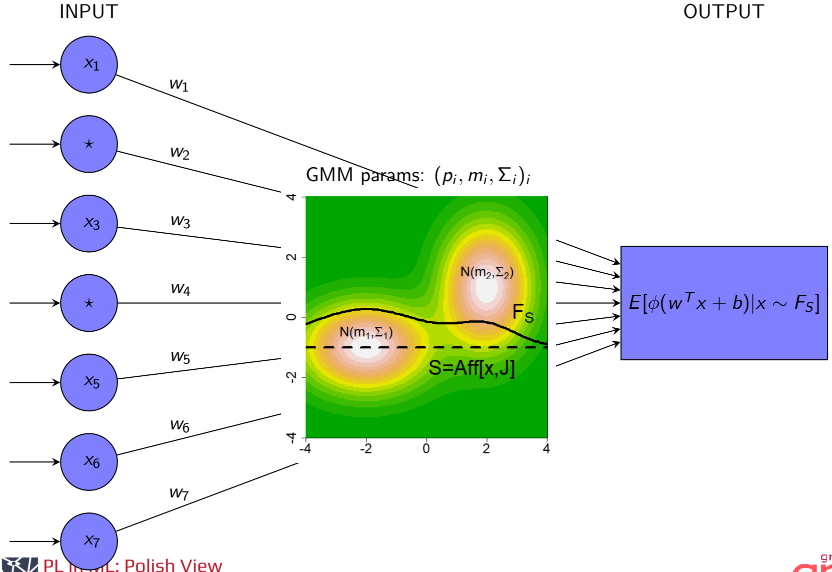
The value of activation function

$$\mathbb{R}^D \ni x \to \phi_\Theta(x) \in \mathbb{R}$$

at the first layer is replaced by

$$E[\phi_\Theta(x)|x \sim F_S].$$

Illustration



Questions

- How to find a density of missing data F?
- How to compute restricted density F_S?
- How to evaluate the expected value of activity functions $E[\phi(x)|x \sim F_S]$?
- What is the learning procedure?

Missing data density

Assumption 1

We assume that $F = \sum_i p_i N(m_i, \Sigma)$, where all $\Sigma_i = \mathrm{diag}(\sigma_1^i, \ldots, \sigma_D^i)$.

- At the beginning, we estimate p_i, m_i, Σ_i from data (using EM),
- We will optimize the above parameters together with network weights,

Representation of missing data 1

Single Gaussian

Let $F = N(m, \Sigma)$, where Σ is diagonal, and $S = \mathrm{Aff}[x, J]$. Then the restricted density equals $F_S = N(m_S, \Sigma_S)$.

Example:

$$x = \begin{pmatrix} ? & ? & -2 & 1 \end{pmatrix} \quad m = \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 \\ 0 & 0 & 13 & 0 \\ 0 & 0 & 0 & 19 \end{pmatrix}$$

$$m_S = \begin{pmatrix} \boxed{1 \quad 2} & -2 & 1 \end{pmatrix} \quad \Sigma_S = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Representation of missing data

Restricted density

Let $F = \sum_i p_i N(m_i, \Sigma_i)$ be the mixture of gaussians, where all $\Sigma_i = \mathrm{diag}(\sigma_1^i, \ldots, \sigma_D^i)$ and let $S = \mathrm{Aff}[x, J]$.
Then the restricted density equals

$$F_S = \sum_i q_i N(m_S^i, \Sigma_S^i),$$

where

$$m_S^i = [x_{J'}, (m_i)_J],$$
$$\Sigma_S^i = [0_{J'J'}, (\Sigma_i)_{JJ}],$$
$$q_i = p_i \text{ , for marginal density...}$$

See the paper for the formulas of regularized restriction – weights p_i are more difficult to compute...

Expected activation

Consider a ReLU function:

$$\mathrm{ReLU}_{w,b}(x) = \max(0, w^T x + b) \text{ , for } w \in \mathbb{R}^D, b \in \mathbb{R}$$

Analytical formula for ReLU

Let $F_S = \sum_i p_i N(m_S^i, \Sigma_S^i)$ be the representation of missing data point. Then

$$\mathrm{ReLU}_{w,b}(F_S) = \sum_i p_i \mathrm{NR}\Big(\frac{w^T m_S^i + b}{\sqrt{w^T \Sigma_S^i w}}\Big),$$

where

$$\mathrm{NR}(w) = \tfrac{1}{\sqrt{2\pi}} \exp(-\tfrac{w^2}{2}) + \tfrac{w}{2}(1 + \mathrm{erf}(\tfrac{w}{\sqrt{2}})),$$
$$\mathrm{erf}(z) = \tfrac{2}{\sqrt{pi}} \int_0^z \exp(-t^2) dt.$$

Basic steps

Initiallize a missing data model $F = \sum_i p_i N(m_i, \Sigma_i)$.

1. Given $S = \mathrm{Aff}[x, J]$ compute F_S
2. At first layer compute $E[\phi(x)|x \sim F_S]$, where ϕ is the activation function
3. Pass this response to subsequent layers
4. Optimize networks parameters together with density model F with use of gradient descent

Parameters of mixture model are trainable – we fit such a density model which minimizes the cost function.

Part II

Processing sets by neural networks

Problem

- Data set X is a family of sets $X_1, \ldots, X_N$
- Every set $X_i \subset \mathbb{R}^D$ is an individual element of a data set.

Question

Typical networks process vectors one by one:

- how to feed the whole set to the network?
- how to obtain a single output for the whole set?

Examples of tasks

We are given a family of sets, where each one has label (X_i, y_i).
Possible tasks:

- Learning the number of clusters (centers of clusters) for the input set.
- Learning a decision boundary for a given set.
- Finding the entropy (or any other statistic) of set.

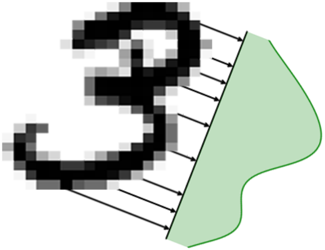
Structured data are usually represented as a set of features, so the set processing is essential.

Idea: projection

Cramer-Wold Theorem

Two sets are equal if they are equal on all one-dimensional projections.

Without loss of information, we can process sets X through their one-dimensional projections $v^T X$, where $v \in \mathbb{R}^D$



Step 1:

Pick some $v_1, \ldots, v_M \in \mathbb{R}^D$ and project X onto them.

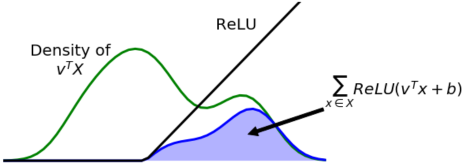
Idea: aggregation

- Let $v^T X \subset \mathbb{R}$ be 1D set and let $b \in \mathbb{R}$ be a fixed bias
- We define aggregated ReLU:

$$\begin{aligned} \mathrm{ReLU}_{v,b}(X) &= \textstyle\sum_{x \in X} \mathrm{ReLU}_{v,b}(x) \\ &= \textstyle\sum_{x \in X} \max(v^T x + b, 0) \in \mathbb{R}. \end{aligned}$$

Fact

We can reconstruct $v^T X \subset \mathbb{R}$ iff we know $\mathrm{ReLU}_{v,b}(X)$ for all $b \in \mathbb{R}$.



Step 2:

Pick some $b_1, \ldots, b_M \in \mathbb{R}$ and compute $\mathrm{ReLU}_{v,b_i}(X)$, for all i.

Procedure

Take a neural network with M output neurons (each parameterized by v, b)

- Project $X \subset \mathbb{R}^D$ onto 1D by $v^T X$
- Apply ReLU for every element

$$\mathrm{ReLU}_{v,b}(x) = \max(v^T x + b, 0)$$

- Summarize the result:

$$\begin{aligned} \mathrm{ReLU}_{v,b}(X) &= \textstyle\sum_{x \in X} \mathrm{ReLU}_{v,b}(x) \\ &= \textstyle\sum_{x \in X} \max(v^T x + b, 0) \end{aligned}$$

Representation:

Set aggregation network (SAN) with M output neurons gives M-dimensional representation of the set.

Generalization

- This procedure works also for other activity functions ϕ_Θ

$$\phi_\Theta(X) = \sum_{x \in X} \phi_\Theta(x)$$

- We can train neuron weights to obtain the most optimal representation.

Fact

If we take a large number of aggregative neurons, where ϕ is universal approximator, then SAN can uniquely identify every input set[3,4].

[3] Ł. Maziarka, et al., *Deep processing of structured data*, arXiv:1810.01868, 2018.
[4] M. Zaheer, et al. *Deep sets*, NIPS 2017

Connection with missing data processing

- For missing data point F_S, the expected activation of ϕ equals:

$$\phi(F_S) = \int \phi(x) F_S(x) dx$$

- For set X, the aggregative neuron ϕ is

$$\phi(X) = \frac{1}{|X|} \sum_{x \in X} \phi(x)$$

This is the expected activation over uniform discrete probability distribution.

Part III

Processing structured data by neural networks.

General view

Let X be structured data, e.g. text, image, graph, etc.

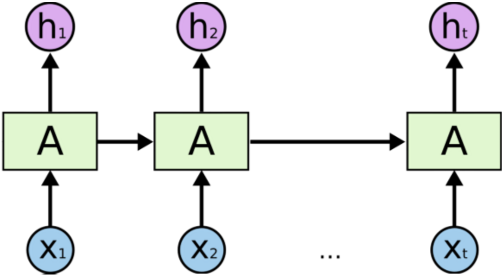
Processing pipeline

$$X = (x_i)_i \stackrel{\Psi}{\to} (\Psi x_i)_i \stackrel{\mathrm{Pool}}{\to} \mathrm{Pool}\{\Psi(x_i) : i\} \stackrel{\Phi}{\to} \mathbb{R}^N.$$

1. Ψ – feature extraction
2. Pool – set aggregation
3. Φ – final output

Step 1a

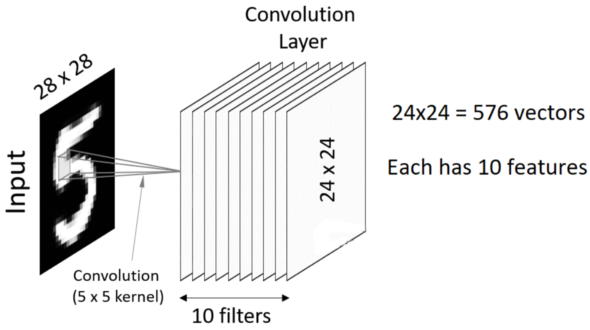
Recurrent network for extracting sequential patterns (e.g. for texts)



ψ returns a sequence of vectors.

Step 1b

Convolutional networks for extracting local patterns (e.g. for images)



ψ returns the image.

Step 2

Let $X \subset \mathbb{R}^K$ be a set of extracted features

- Typically, take max over each attribute to produce a fixed length vector
- Better idea is to replace pooling layer by set aggregation network (SAN) to preserve necessary information from a set.

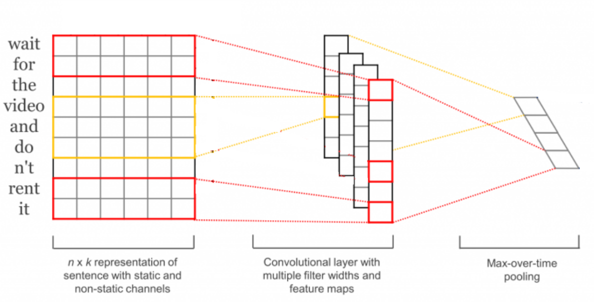


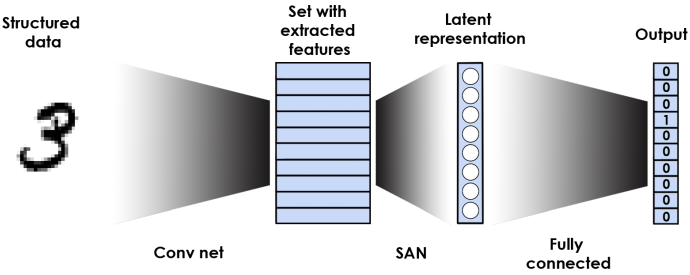
Figure: Max-pooling for 1D-convolutions.

Step 3

Let $x \in \mathbb{R}^K$ be an aggregated vector.

- Use a classical (fully connected) neural network ϕ to produce a final output

Summary


Structured data
Set with extracted features
Latent representation
Output
0
0
0
1
0
0
0
0
0
0
Conv net
SAN
Fully connected