Multi-output machine learning with applications to genomics

Institut Imagine

Ekaterina Antonenko

May 26, 2023

Laboratoire d'informatique, École Polytechnique, IP Paris

Table of contents

About me

About me

Education

since 2020 PhD candidate: École Polytechnique
Data Science and Mining (DaSciM) team, Laboratoire d'informatique (LIX), CIFRE with Digitalent, Scientific advisor: Jesse Read

2012–2014 2-year program: Moscow Bioinformatics School

2009–2014 Diploma in Mathematics: Moscow State University

Employment

since 2020 École Polytechnique: Researcher, Teaching assistant (Machine learning bachelor course)

2017–2019 Math instructor, New York, USA

2013–2016 Analyst, Moscow, Russia

About me

Keywords

Multi-output prediction; Missing value imputation; Explainability and interpretability; Tree methods; Probabilisctical inference.

Papers

pre-print E. Antonenko, R. Beigaitė, M. Mechenich, J. Read and I. Žliobaitė, *Backward inference in probabilistic Regressor Chains with distributional constraints.*

pre-print E. Antonenko, A.Carreño, J. Read, *Autoreplicative Random Forests for missing value imputation.*

pre-print M. Konnova, E. Antonenko, J. Read, *Missing value imputation for genomics data using a Sequence Based Generative Adversarial Network (SBGAN).*

ECML 2022 E. Antonenko, J. Read, *Chains of Autoreplicative Random Forests for missing value imputation in high-dimensional datasets*, [Best paper award].

IDA 2022 E. Antonenko, J. Read, *Multi-modal ensembles of regressor chains for multi-output prediction*, Advances in Intelligent Data Analysis XXI - 21st International Symposium.

PeerJ 2016 V. Ivanenko, E. Antonenko, M. Gelfand, J. Yager, F. Ferrari, *Changes in segmentation and setation along the anterior/posterior axis of the homonomous trunk limbs of a remipede (Crustacea, Arthropoda).*

Introduction: Multi-output prediction

Definition of a multi-output problem

Given: Dataset $\mathcal{D} = \{(\boldsymbol{x}^i, \boldsymbol{y}^i)\}_{i=1}^N$ of N samples:

- features $\boldsymbol{x}^i = [x_1^i, ..., x_M^i]$
- outputs $\boldsymbol{y}^i = [y_1^i, ..., y_L^i]$

Goal: Model $f(\boldsymbol{X}) = \boldsymbol{y}$ which outputs predictions $\hat{\boldsymbol{y}}^i = [\hat{y}_1^i, ..., \hat{y}_L^i]$ having $\mathcal{D}$ observed.

Definition of a multi-output problem

Given: Dataset $\mathcal{D} = \{(\boldsymbol{x}^i, \boldsymbol{y}^i)\}_{i=1}^N$ of N samples:

- features $\boldsymbol{x}^i = [x_1^i, ..., x_M^i]$
- outputs $\boldsymbol{y}^i = [y_1^i, ..., y_L^i]$

Goal: Model $f(\boldsymbol{X}) = \boldsymbol{y}$ which outputs predictions $\hat{\boldsymbol{y}}^i = [\hat{y}_1^i, ..., \hat{y}_L^i]$ having $\mathcal{D}$ observed.

Example:

	X	Age	Height	Body length	Weight
A	· · ·	12,44	127,4	151	294,5
B	· · ·	9,44	137,6	156	328
C	· · ·	10,44	128,6	157	377
D	· · ·	6,13	125,6	150	305,5
E	· · ·	6,15	139	156	325
F	· · ·	?	?	?	?

Definition of a multi-output problem

Given: Dataset $\mathcal{D} = \{(\boldsymbol{x}^i, \boldsymbol{y}^i)\}_{i=1}^N$ of N samples:

- features $\boldsymbol{x}^i = [x_1^i, ..., x_M^i]$
- outputs $\boldsymbol{y}^i = [y_1^i, ..., y_L^i]$

Goal: Model $f(\boldsymbol{X}) = \boldsymbol{y}$ which outputs predictions $\hat{\boldsymbol{y}}^i = [\hat{y}_1^i, ..., \hat{y}_L^i]$ having $\mathcal{D}$ observed.

Example:

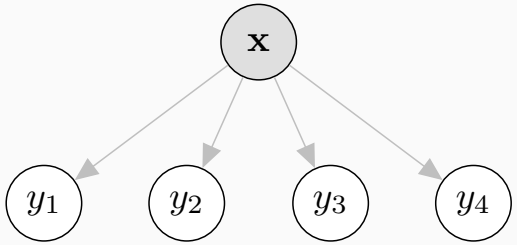
	x	Age	Height	Body length	Weight
A	· · ·	12,44	127,4	151	294,5
B	· · ·	9,44	137,6	156	328
C	· · ·	10,44	128,6	157	377
D	· · ·	6,13	125,6	150	305,5
E	· · ·	6,15	139	156	325
F	· · ·	?	?	?	?

Idea: to model these labels together in order to get better prediction performance

Some approaches to multi-output problems

- **Independent models**

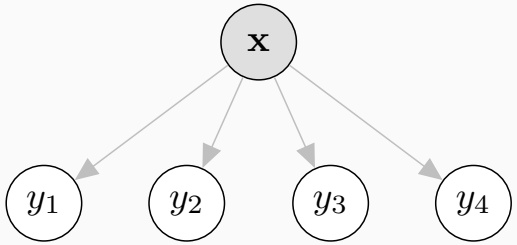
$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), ..., h_L(\boldsymbol{x})]$$



Some approaches to multi-output problems

- **Independent models**

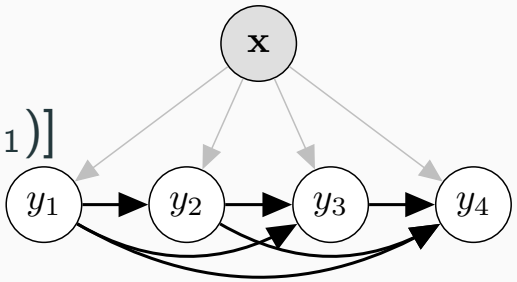
$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), ..., h_L(\boldsymbol{x})]$$



- **Fully-cascaded chain**

$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, \hat{y}_1), ..., h_L(\boldsymbol{x}, \hat{y}_1, ..., \hat{y}_{L-1})]$$

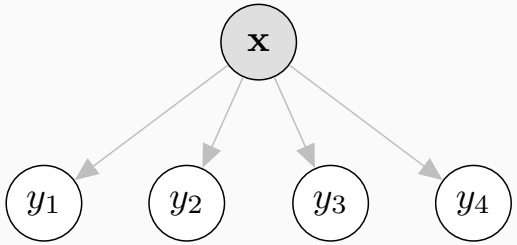
$h_1, ..., h_L$ = any single-output models



Some approaches to multi-output problems

- **Independent models**

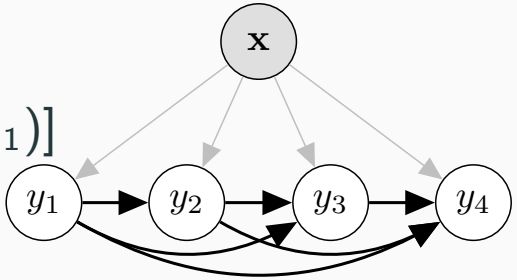
$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), ..., h_L(\boldsymbol{x})]$$



- **Fully-cascaded chain**

$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, \hat{y}_1), ..., h_L(\boldsymbol{x}, \hat{y}_1, ..., \hat{y}_{L-1})]$$

$h_1, ..., h_L$ = any single-output models

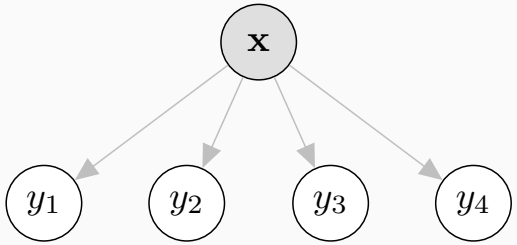


	x	y_1	y_2	y_3	y_4
F	· · ·	12,32	?	?	?

Some approaches to multi-output problems

- **Independent models**

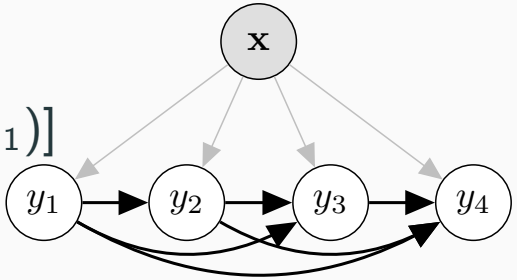
$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), ..., h_L(\boldsymbol{x})]$$



- **Fully-cascaded chain**

$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, \hat{y}_1), ..., h_L(\boldsymbol{x}, \hat{y}_1, ..., \hat{y}_{L-1})]$$

$h_1, ..., h_L$ = any single-output models



	x	y_1	y_2	y_3	y_4
F	· · ·	12,32	125,1	?	?

- **Multi-output Decision Trees and Random Forests**

$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [\boldsymbol{h}(\boldsymbol{x})]$$

PICTURE

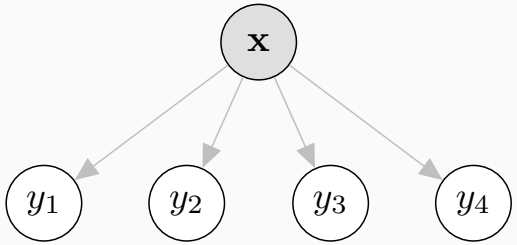
all labels are assigned simultaneously

BUT the metric to optimize is still decomposable

Some approaches to multi-output problems

- **Independent models**

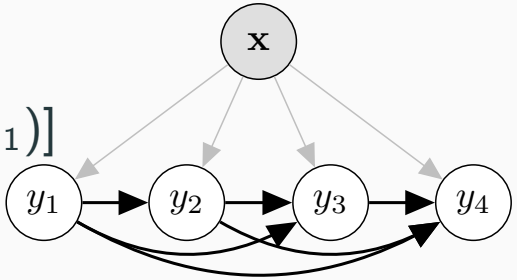
$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), ..., h_L(\boldsymbol{x})]$$



- **Fully-cascaded chain**

$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, \hat{y}_1), ..., h_L(\boldsymbol{x}, \hat{y}_1, ..., \hat{y}_{L-1})]$$

$h_1, ..., h_L$ = any single-output models



	x	y_1	y_2	y_3	y_4
F	· · ·	12,32	125,1	143	?

- **Multi-output Decision Trees and Random Forests**

$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [\boldsymbol{h}(\boldsymbol{x})]$$

PICTURE

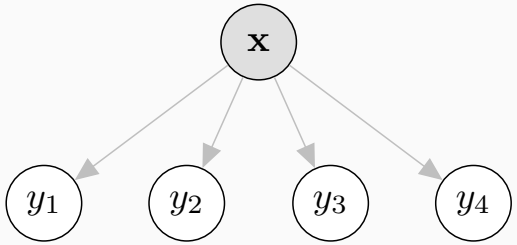
all labels are assigned simultaneously

BUT the metric to optimize is still decomposable

Some approaches to multi-output problems

- **Independent models**

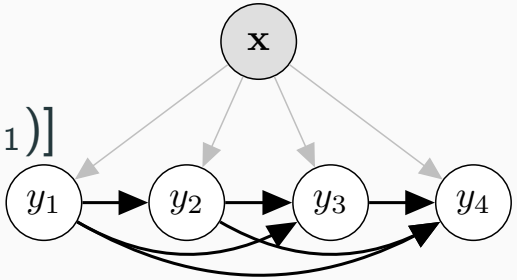
$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), ..., h_L(\boldsymbol{x})]$$



- **Fully-cascaded chain**

$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, \hat{y}_1), ..., h_L(\boldsymbol{x}, \hat{y}_1, ..., \hat{y}_{L-1})]$$

$h_1, ..., h_L$ = any single-output models



	x	y_1	y_2	y_3	y_4
F	· · ·	12,32	125,1	143	289,3

- **Multi-output Decision Trees and Random Forests**

$$\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_L] = [\boldsymbol{h}(\boldsymbol{x})]$$

PICTURE

all labels are assigned simultaneously

BUT the metric to optimize is still decomposable

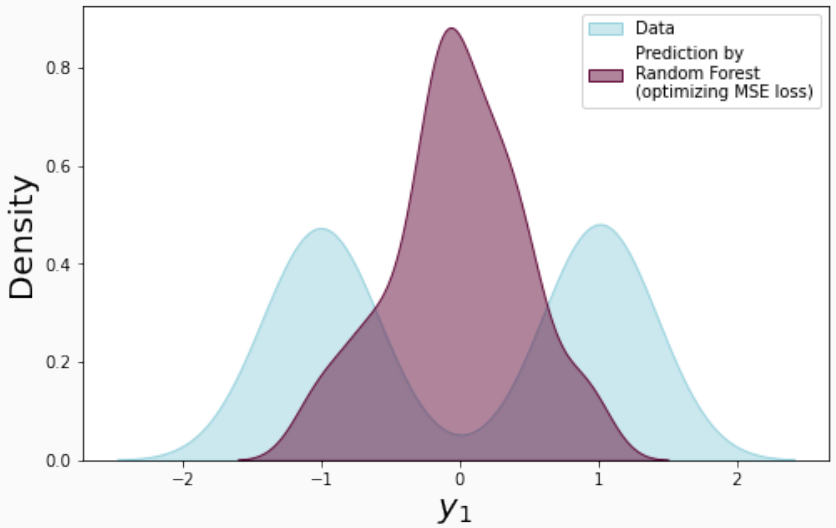
Regression: some problems

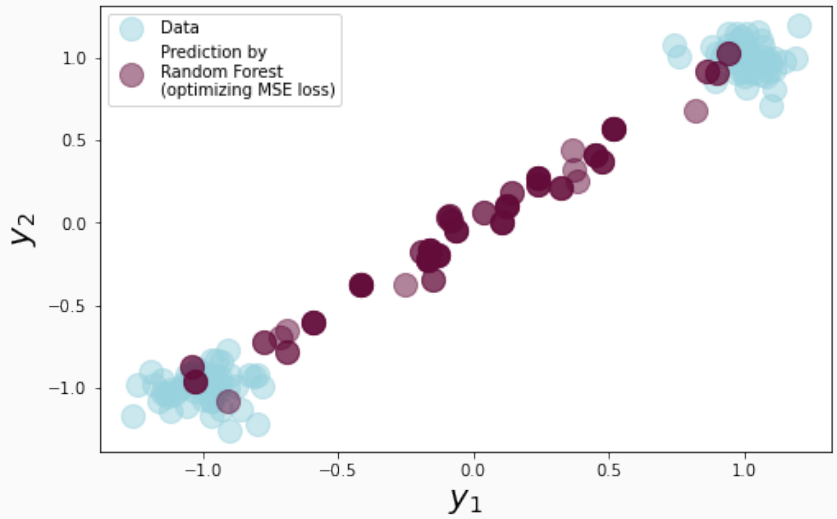
Problem 1. Multi-modal distributions

Multi-output models may not work well for multi-modal distributions.

One possible reason: inadequate choice of the loss function.

Most models optimize MSE $= \frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j)^2$.





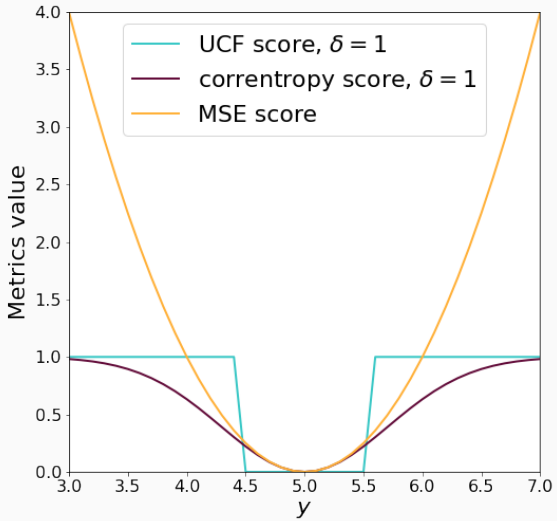
Optimizing MSE does not help to exploit the dependencies between the targets.

Problem 1. Solution

Uniform Cost Function (UCF) is an analogue of 0/1 loss for regression.

$$\mathrm{UCF}(\delta) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}0 \text{ if } \|\boldsymbol{y}^i - \hat{\boldsymbol{y}}^i\|_2 < \frac{\delta}{2},\\ 1 \text{ otherwise.}\end{cases}$$

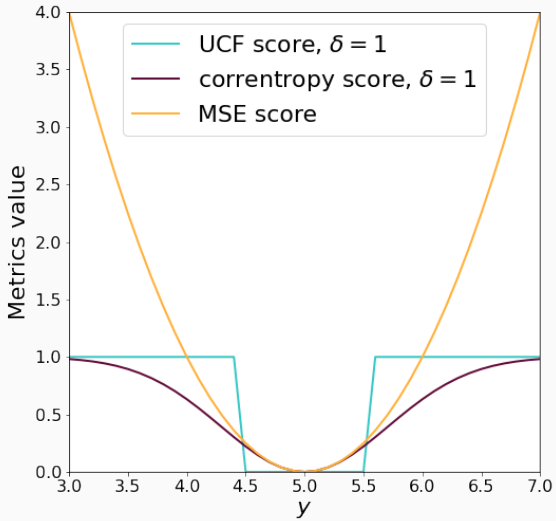
Correntropy is a smooth version of UCF.
FORMULA



Problem 1. Solution

Uniform Cost Function (UCF) is an analogue of 0/1 loss for regression.

$$\mathrm{UCF}(\delta) = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 0 \text{ if } \|\boldsymbol{y}^i - \hat{\boldsymbol{y}}^i\|_2 < \frac{\delta}{2}, \\ 1 \text{ otherwise.} \end{cases}$$



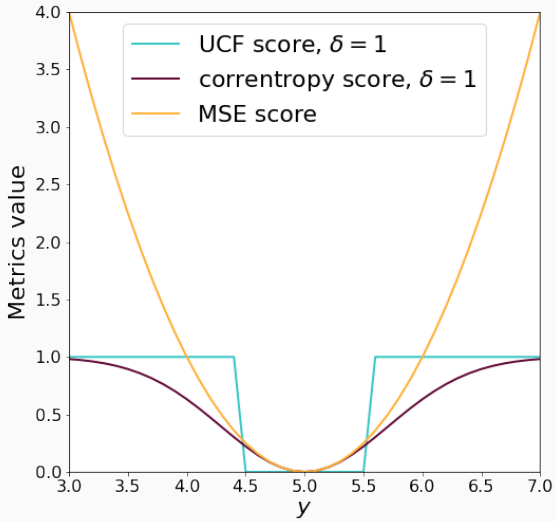
Correntropy is a smooth version of UCF.
FORMULA

Goal = challenge: optimize UCF or correntropy in Regressor Chains with *any base estimator*.

Problem 1. Solution

Uniform Cost Function (UCF) is an analogue of 0/1 loss for regression.

$$\mathrm{UCF}(\delta) = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 0 \text{ if } \|\boldsymbol{y}^i - \hat{\boldsymbol{y}}^i\|_2 < \frac{\delta}{2}, \\ 1 \text{ otherwise.} \end{cases}$$



Correntropy is a smooth version of UCF.
FORMULA

Goal = challenge: optimize UCF or correntropy in Regressor Chains with *any base estimator*.

✓

Problem 2. Backward inference in Regressor Chains

Problem 2. Solution

Classification: missing value imputation

Missing data

Why is data missing?

- Errors in sensors
- Human factor (reluctance to answer particular questions)
- Combining different studies
- ...

Missing data

Why is data missing?

- Errors in sensors
- Human factor (reluctance to answer particular questions)
- Combining different studies
- ...

Why to impute the missing data?

- Most off-the-shelf statistical and machine learning methods cannot handle missing values
- Considering only instances with complete information can lead to a loss of necessary information and can yield a very poor or even empty dataset
- Missing data itself might be of interest

Missing data

Why is data missing?

- Errors in sensors
- Human factor (reluctance to answer particular questions)
- Combining different studies
- ...

Why to impute the missing data?

- Most off-the-shelf statistical and machine learning methods cannot handle missing values
- Considering only instances with complete information can lead to a loss of necessary information and can yield a very poor or even empty dataset
- Missing data itself might be of interest

Types of missingness

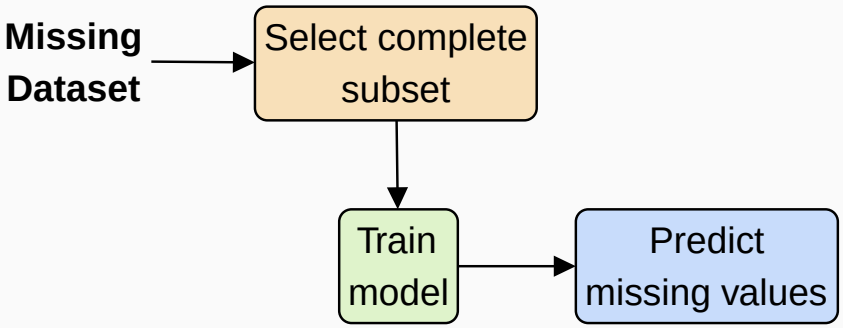
- Missing Completely at Random (MCAR): entirely independently of feature values
- Missing at Random (MAR): depends only on the observed feature values
- Missing Not at Random (MNAR): depends on both the observed and the unobserved feature values

Procedural and Iterative Imputation

Procedural

One procedure:

- Train a model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed

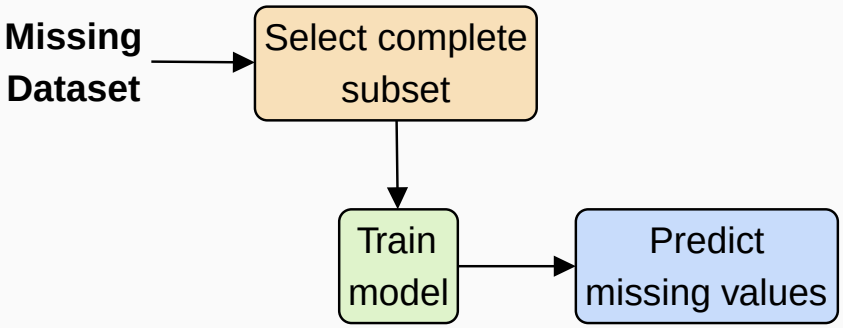


Procedural and Iterative Imputation

Procedural

One procedure:

- Train a model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed



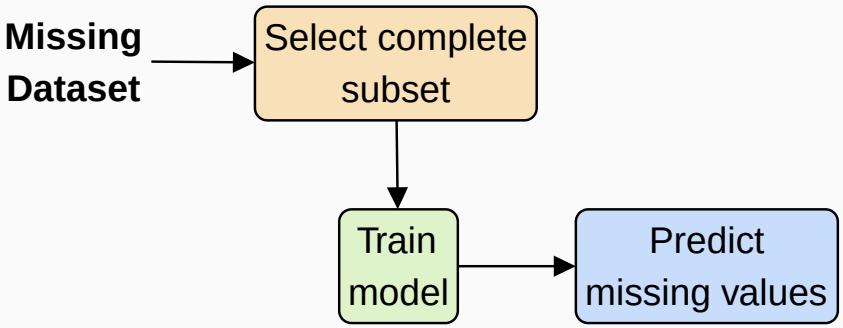
Examples: mode, kNN, PCA, Autoencoders

Procedural and Iterative Imputation

Procedural

One procedure:

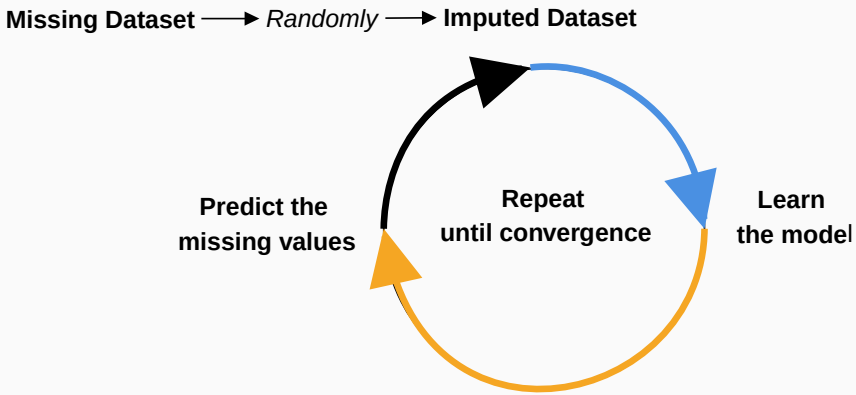
- Train a model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed



Iterative

Impute randomly, then repeat:

- Train a model on previous imputation
- Use the fitted model to predict on all instances
- Correct observed values if changed



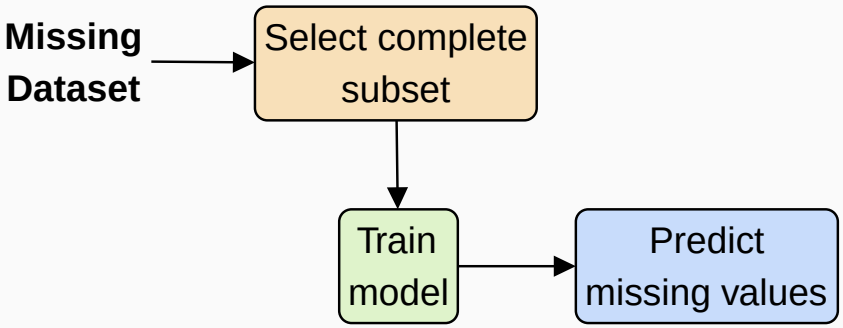
Examples: mode, kNN, PCA, Autoencoders

Procedural and Iterative Imputation

Procedural

One procedure:

- Train a model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed

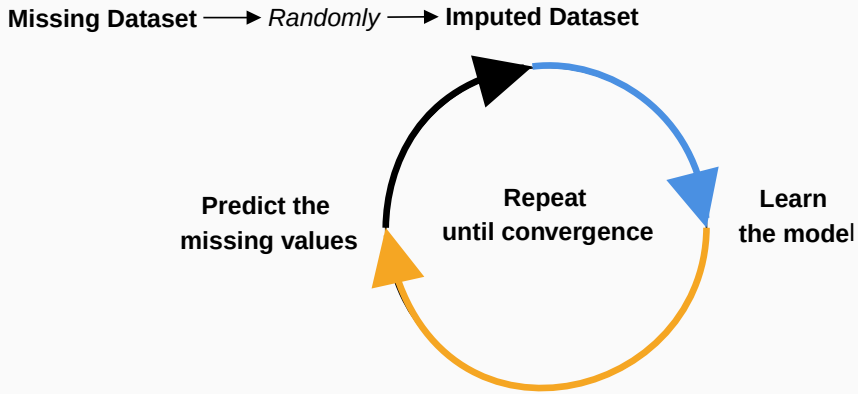


Examples: mode, kNN, PCA, Autoencoders

Iterative

Impute randomly, then repeat:

- Train a model on previous imputation
- Use the fitted model to predict on all instances
- Correct observed values if changed



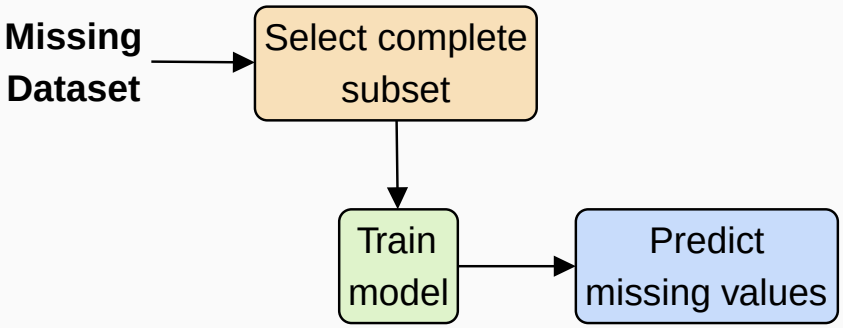
Examples: MICE, MissForest, PCA, Autoencoders

Procedural and Iterative Imputation

Procedural

One procedure:

- Train a model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed

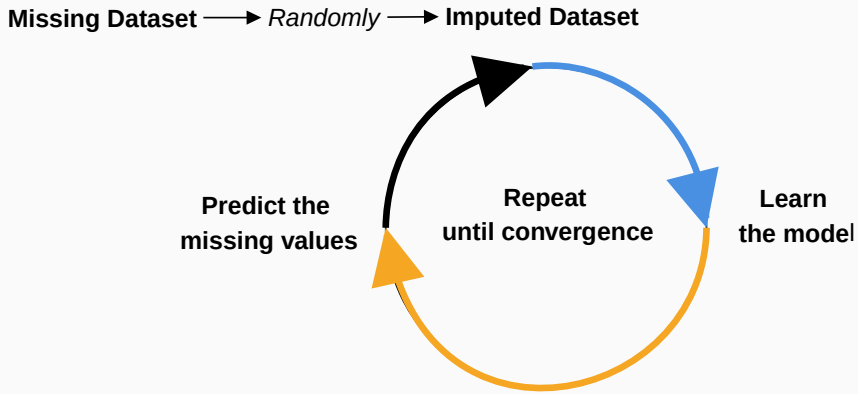


Examples: mode, kNN, PCA, Autoencoders

Iterative

Impute randomly, then repeat:

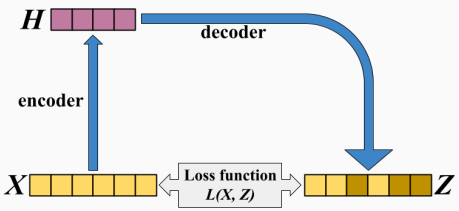
- Train a model on previous imputation
- Use the fitted model to predict on all instances
- Correct observed values if changed



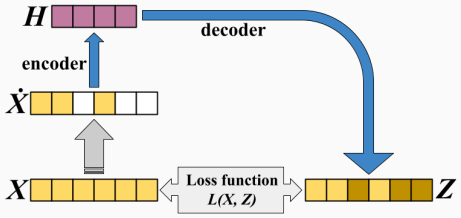
Examples: MICE, MissForest, PCA, Autoencoders

+ Autoreplicative Random Forests

Autoencoders $\Longrightarrow$ Autoreplicative Random Forests

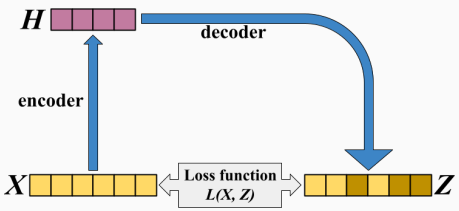


Autoencoder

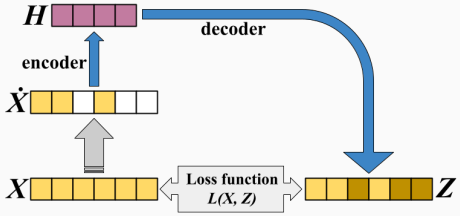


Denoising Autoencoder

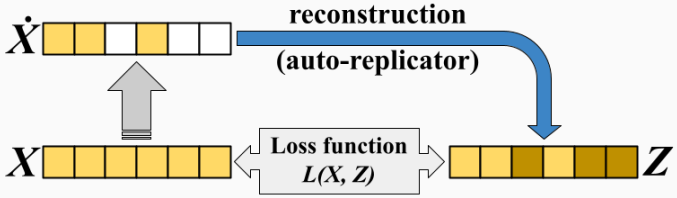
Autoencoders $\Longrightarrow$ Autoreplicative Random Forests



Autoencoder

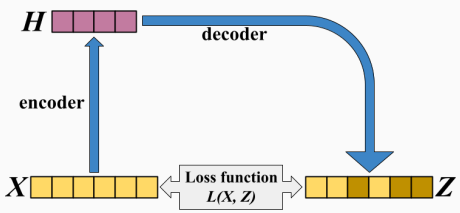


Denoising Autoencoder

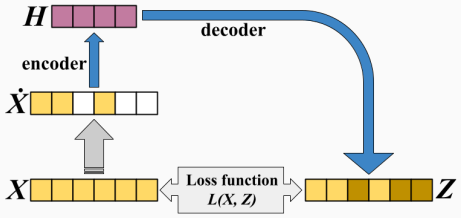


Autoreplicator

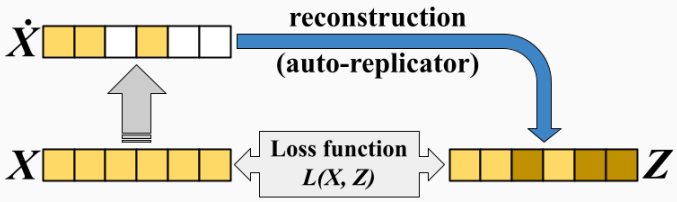
Autoencoders $\Longrightarrow$ Autoreplicative Random Forests



Autoencoder



Denoising Autoencoder

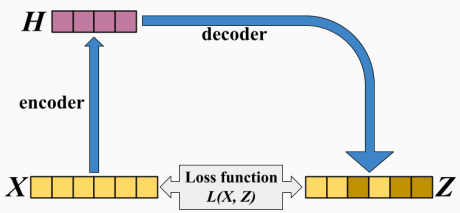


Autoreplicator

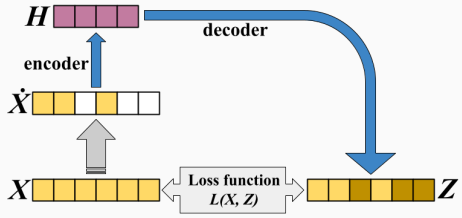
Why to use multi-label methods?

- compared to one-by-one methods: may deeper exploit interdependencies between the targets + less computationally expensive
- compared to neural networks: fewer parameters (good for low-sampled data)
- no need for hidden layers

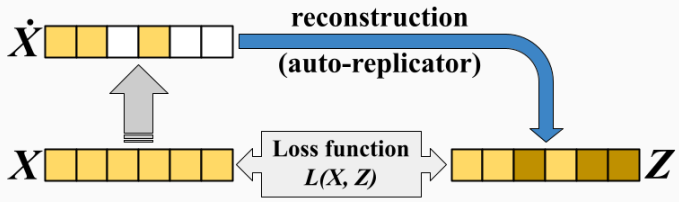
Autoencoders $\Longrightarrow$ Autoreplicative Random Forests



Autoencoder



Denoising Autoencoder



Autoreplicator

Why to use multi-label methods?

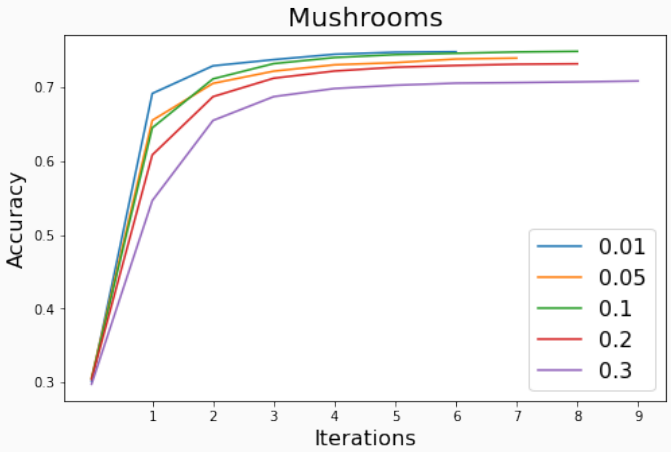
- compared to one-by-one methods: may deeper exploit interdependencies between the targets + less computationally expensive
- compared to neural networks: fewer parameters (good for low-sampled data)
- no need for hidden layers

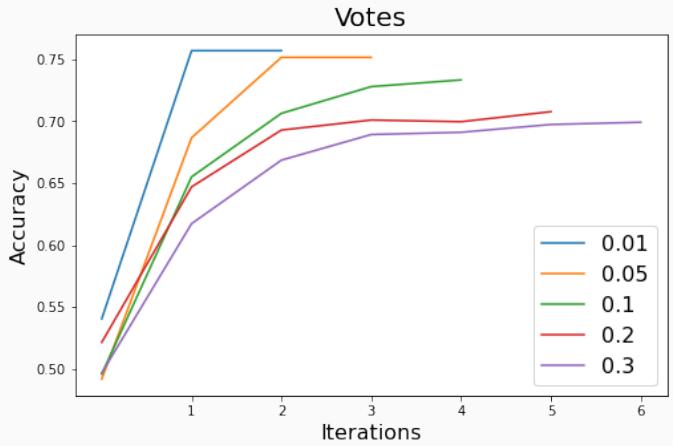
Which methods?

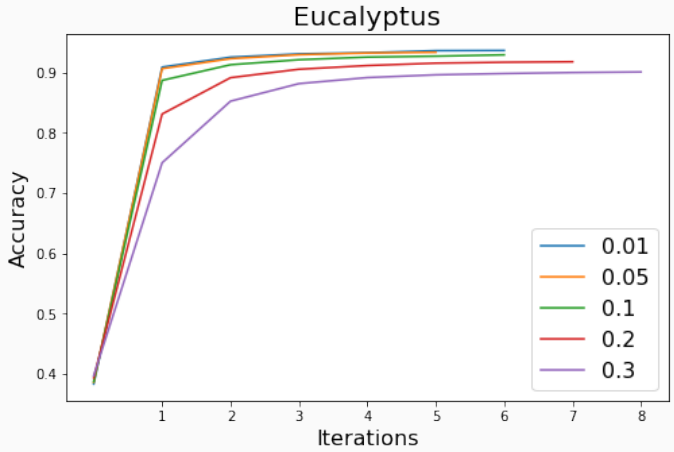
- Decision Trees, Random Forests, Extra Trees
- Classifier Chains, Multilabel k Nearest Neighbours, Random k-Labelsets, Conditional Dependency Networks, etc.

Results: iterative ARF do converge

Imputation via Iterative Random Forests converges after several iterations







Accuracy of imputation

MVR	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3
	Mushroom [8,124 x 22]					Soybean [307 x 35]					Tumor [339 x 17]				
Complete cases	80.1%	32.3%	10.1%	0.7%	0.04%	69.7%	13.7%	1.0%	0%	0%	83.8%	38.9%	15.0%	1.2%	0.3%
MICE	0.658	0.715	0.741	**0.769**	**0.777**	**0.884**	**0.884**	**0.879**	**0.867**	**0.850**	**0.761**	**0.768**	**0.748**	**0.754**	**0.735**
itARF	0.730	0.740	0.747	0.734	0.707	0.824	0.850	0.832	0.815	0.789	0.652	0.672	0.645	0.660	0.620
pARF	**0.748**	**0.774**	**0.761**	0.671	0.478	0.804	0.779	0.600	–	–	0.639	0.696	0.650	0.694	0.635
itAE	0.608	0.618	0.604	0.584	0.569	0.653	0.607	0.608	0.584	0.590	0.721	0.732	0.692	0.711	0.710
pAE	0.580	0.494	0.491	0.538	0.428	0.653	0.622	0.594	–	–	0.721	0.718	0.692	0.690	0.497
itPCA	0.604	0.627	0.622	0.623	0.618	0.667	0.692	0.671	0.646	0.603	0.721	0.740	0.692	0.711	0.710
pPCA	0.600	0.587	0.578	0.537	0.441	0.655	0.639	0.620	–	–	0.721	0.671	0.688	0.626	0.411

	Votes [435 x 16]					Lymphography [148 x 18]					Financial Survey [6,394 x 212]				
Complete cases	85.3%	42.2%	18.5%	1.3%	0.4%	81.8%	40.5%	14.9%	2.7%	0%	11.8%	0%	0%	0%	0%
MICE	**0.768**	**0.795**	**0.771**	**0.768**	**0.782**	**0.750**	**0.679**	**0.665**	**0.648**	**0.651**	–	–	–	–	–
itARF	0.719	0.726	0.728	0.723	0.718	0.714	0.639	0.638	0.628	0.600	**0.684**	**0.677**	**0.676**	**0.667**	**0.661**
pARF	0.730	0.758	0.756	0.522	0.495	0.636	0.647	0.604	0.608	–	0.633	–	–	–	–
itAE	0.697	0.563	0.602	0.578	0.570	0.700	0.474	0.485	0.448	0.487	0.626	0.617	0.616	0.604	0.596
pAE	0.638	0.546	0.600	0.524	0.488	0.679	0.514	0.563	0.611	–	0.313	–	–	–	–
itPCA	0.665	0.583	0.567	0.572	0.570	0.686	0.513	0.477	0.468	0.484	0.653	0.645	0.645	0.634	0.627
pPCA	0.595	0.499	0.567	0.507	0.453	0.693	0.536	0.562	0.502	–	0.299	–	–	–	–

- Procedural ARFs: may be powerful when enough complete instances
- MICE: as powerful as computationally expensive
- Iterative ARFs: still powerful + significantly quicker

Time complexity

In theory:

- p = number of features
- n_{iter} = number of iterations

itARF	pARF	MICE
$\mathcal{O}(n_{iter} \cdot p)$	$\mathcal{O}(p)$	$\mathcal{O}(n_{iter} \cdot p^2)$

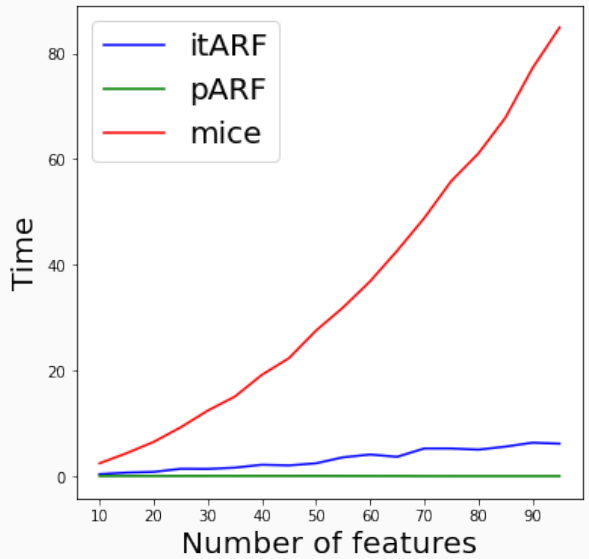
Time complexity

In theory:

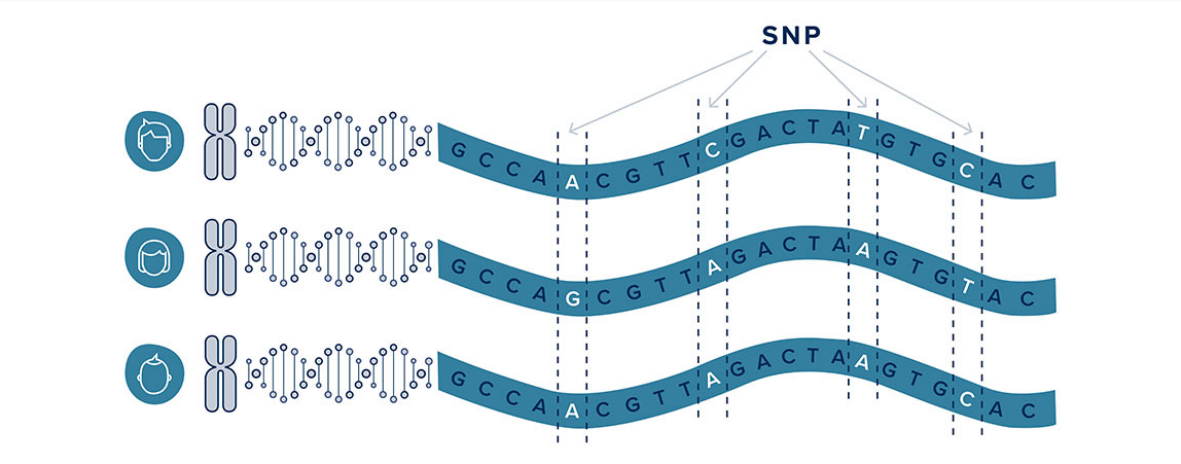
- p = number of features
- n_{iter} = number of iterations

itARF	pARF	MICE
$\mathcal{O}(n_{iter} \cdot p)$	$\mathcal{O}(p)$	$\mathcal{O}(n_{iter} \cdot p^2)$

In practice:

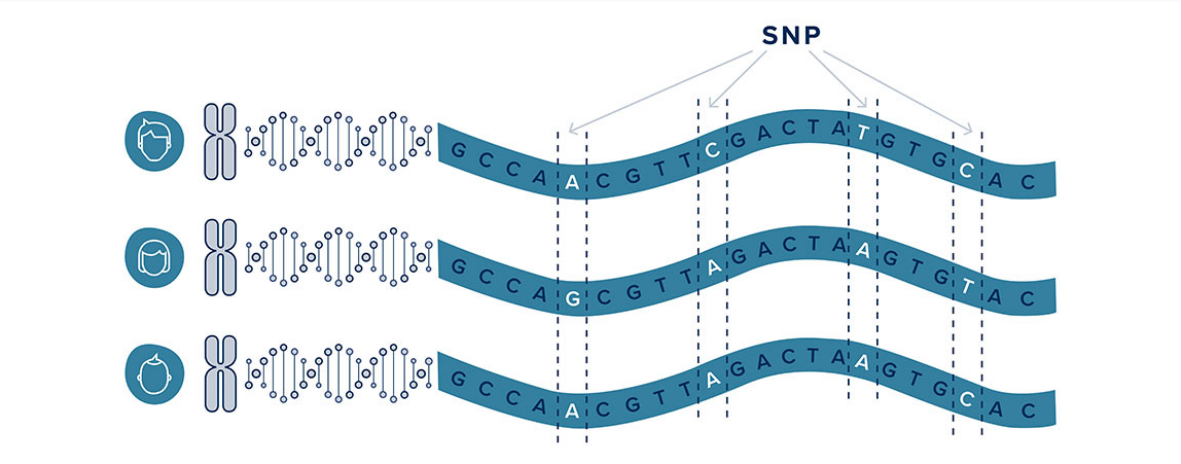


Usecase: Single Nucleotide Polymorphisms (SNP)



- Categorical: 0 (dominant-dominant), 1 (dominant-mutant), 2 (mutant-mutant)
- High-dimensional ($10^5 - 10^6$) and low-sampled ($10^2 - 10^3$)
- Ordering is important
- Missing values occur due to external mechanisms $\implies$ MCAR

Usecase: Single Nucleotide Polymorphisms (SNP)

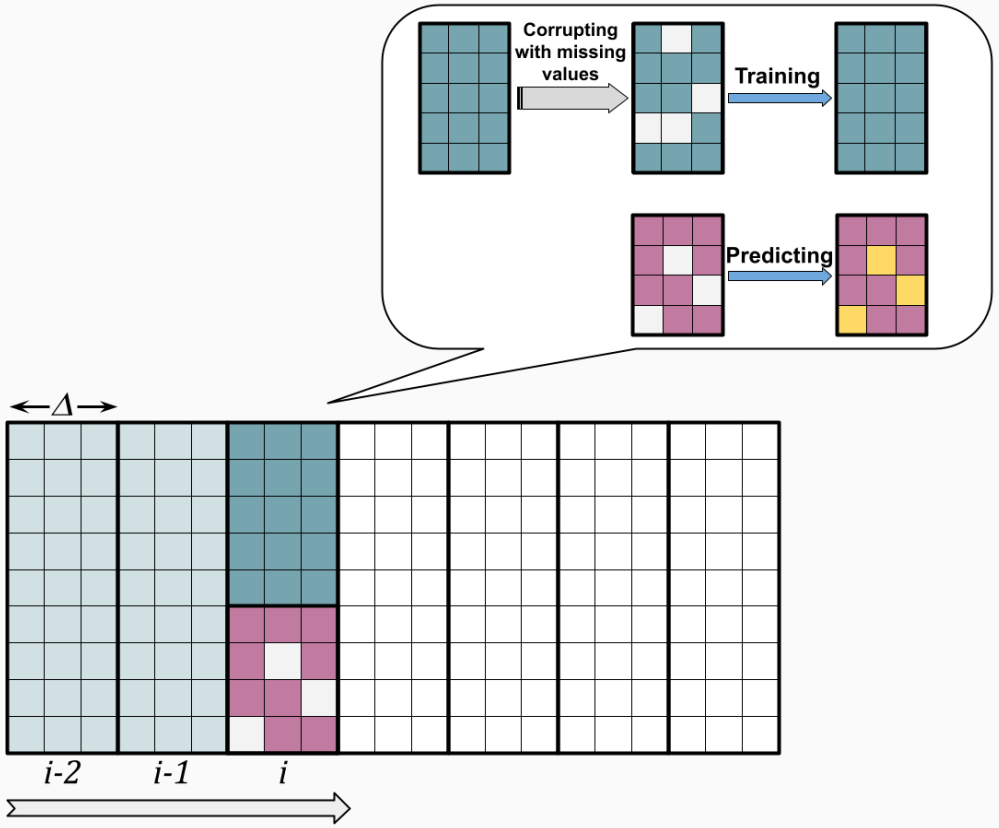


- Categorical: 0 (dominant-dominant), 1 (dominant-mutant), 2 (mutant-mutant)
- High-dimensional ($10^5 - 10^6$) and low-sampled ($10^2 - 10^3$)
- Ordering is important
- Missing values occur due to external mechanisms $\implies$ MCAR

Methods:

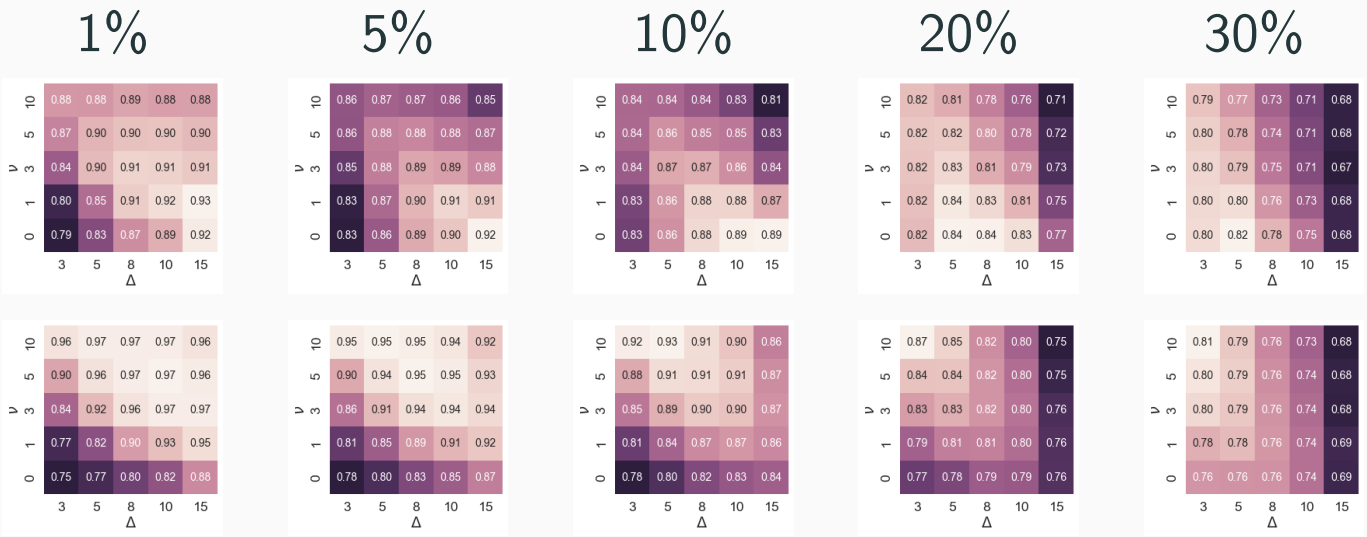
- reference-based (state-of-the-art for human data)
- reference-free (when reference panels are not available)

Chains of Autoreplicative Random Forests



- **Procedural approach:** one window of size $\Delta =$ complete instances $+$ instances with missing values
- **Chain of windows:** on each step, stacking ν windows with already imputed values as additional features
- **Ensemble of chains:** one forward chain, one backward chain, several random chains

Gridsearch for parameters Δ and ν



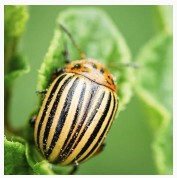
Lighter color / higher accuracy

Δ: bigger fraction of missing values $\rightarrow$ smaller size of window $\Longrightarrow$ can be estimated theoretically, no need for search

ν: may depend on problem

Accuracy

	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3
	Maize [247 x 44,729]					Eucalyptus [970 x 33,398]				
ChARF	**0.952**	**0.935**	**0.916**	**0.882**	**0.845**	**0.970**	**0.950**	**0.926**	**0.866**	0.810
kNN (5/10)	0.803	0.802	0.801	0.798	0.794	0.851	0.849	0.847	0.843	**0.839**
mode	0.727	0.727	0.726	0.727	0.726	0.725	0.732	0.731	0.730	0.729
SVD (50/500)	0.647	0.648	0.645	0.643	0.636	0.788	0.788	0.788	0.785	0.780
MICE	–	–	–	–	–	–	–	–	–	–



	Colorado Beetle [188 x 34,186]					Arabica Coffee [596 x 4,666]				
ChARF	**0.835**	**0.824**	**0.818**	**0.805**	**0.792**	**0.897**	**0.886**	**0.878**	**0.866**	0.854
kNN (50/10)	0.765	0.763	0.765	0.765	0.764	0.867	0.866	0.866	0.865	**0.864**
mode	0.761	0.760	0.762	0.761	0.761	0.807	0.804	0.805	0.805	0.804
SVD (50/100)	0.740	0.737	0.737	0.735	0.734	0.693	0.694	0.696	0.692	0.690
MICE	–	–	–	–	–	0.757	0.741	0.724	0.689	0.664

	Wheat [388 x 9,763]					Coffea Canephora [119 x 45,748]				
ChARF	0.821	0.808	0.795	0.777	0.762	**0.799**	**0.781**	**0.761**	0.731	0.717
kNN (10/10)	**0.823**	**0.819**	**0.818**	**0.815**	**0.811**	0.737	0.739	0.737	**0.734**	**0.731**
mode	0.729	0.727	0.729	0.729	0.727	0.691	0.693	0.692	0.692	0.691
SVD (200/50)	0.622	0.618	0.609	0.600	0.594	0.456	0.453	0.450	0.449	0.450
MICE	0.641	0.635	0.621	0.585	0.545	–	–	–	–	–

- MICE: run with 10 neighbors for each feature, still worked only for smaller data
- Autoencoders: not taken into comparison (no complete data for training)
- Well-known methods for SNP imputation: *k* Nearest Neighbors, Single Value Decomposition

Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection

Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection
- We show how probabilistic training can be easily added to the model

Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection
- We show how probabilistic training can be easily added to the model
- ARF vs MICE: high quality and much faster

Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection
- We show how probabilistic training can be easily added to the model
- ARF vs MICE: high quality and much faster
- ARF vs Autoencoders:
  - no need for one-hot encoding $\implies$ less features.
  - lower time complexity $\implies$ works for high-dimensional datasets
  - no need for complete data

Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?

Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets ($> 10,000$ features) multi-target methods are very memory expensive

Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets ($> 10,000$ features) multi-target methods are very memory expensive
- Studies for MAR and MNAR scenarios

Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets ($> 10,000$ features) multi-target methods are very memory expensive
- Studies for MAR and MNAR scenarios
- Regression (e.g. gene expression)

Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets ($> 10,000$ features) multi-target methods are very memory expensive
- Studies for MAR and MNAR scenarios
- Regression (e.g. gene expression)
- How can we avoid overfitting in Iterative RF?

Thank you!